ROUGH SET BASED SPLITTING CRITERION FOR BINARY DECISION TREE CLASSIFIERS

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DARIUSZ G. MIKULSKI

OAKLAND UNIVERSITY

2006

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		

1. REPORT DATE **26 SEP 2006**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Rough Set Based Splitting Criterion for Binary Decision Tree Classifiers**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Dariusz G. Mikulski**		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **US Army RDECOM-TARDEC 6501 E 11 Mile Rd Warren, MI 48397-5000**		8. PERFORMING ORGANIZATION REPORT NUMBER **16334**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) **TACOM/TARDEC**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) **16334**
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT **SAR**	18. NUMBER OF PAGES **181**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ROUGH SET BASED SPLITTING CRITERION FOR
BINARY DECISION TREE CLASSIFIERS

by

DARIUSZ G. MIKULSKI

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN
COMPUTER SCIENCE AND ENGINEERING

2006

Oakland University
Rochester, Michigan

APPROVED BY:

Djamel Bouchaffra, Ph.D., Chair	Date
Subramaniam Ganesan, Ph.D.	Date
Jia Li, Ph.D.	Date

To my family

ACKNOWLEDGMENTS

ABSTRACT

ROUGH SET BASED SPLITTING CRITERION FOR
BINARY DECISION TREE CLASSIFIERS

by

Dariusz G. Mikulski

Advisor: Djamel Bouchaffra, Ph.D.

Pattern recognition applications often use inductive reasoning to find hidden

relationships and concepts within a data set. Many computational models and heuristics

exist to assist in induction [12, 14, 18, 22, 26, 37]. Even so, researchers continue to

actively develop new inductive methods [8, 10, 45, 46]. The research in this thesis

advances induction for pattern classification by presenting the derivation and

application of a new measure of information based on rough set theory – the *rough

product*. The rough product helps us to understand the manner in which an attribute

value partition affects the upper approximation for each decision class. The thesis also

presents an application of the rough product in a splitting criterion for binary decision

tree classifiers. Using a MATLAB® software tool that we developed for this research,

we compare the performance of the Gini Index, Twoing Rule, Maximum Deviance

Reduction, and Rough Product splitting criteria on various data sets using *k*-folds cross

validation. We determine performance by measuring the following metrics: accuracy,

error rate, precision, recall, F-measure, node count, depth count, and complexity. Our

results suggest that, in the presence of noisy data, the Rough Product splitting criterion

could construct binary decision trees that are simpler and shorter than those produced

by the Gini Index, Twoing Rule, or Maximum Deviance Reduction splitting criteria.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CART   Classification and Regression Tree

CDC   Center for Disease Control and Prevention

FN   False Negative

FP   False Positive

MDL   Minimum Description Length

MDR   Maximum Deviance Reduction

MMA   Mean of Mean Accuracy

MMC   Mean of Mean Complexity

MMDC   Mean of Mean Depth Count

MMER   Mean of Mean Error Rate

MMFM   Mean of Mean F-Measure

MMNC   Mean of Mean Node Count

MMP   Mean of Mean Precision

MMR   Mean of Mean Recall

NP   Nondeterministic Polynomial

NPCR   National Program of Cancer Registries

P   Polynomial

PASH   Parameterized Average Support Heuristic

TN   True Negative

TP   True Positive

WONDER   Wide-ranging ONline Data for Epidemiologic Research

DATA ON COMPACT DISC

Excel Data Sets

1. Breast Cancer Wisconsin.xls

2. Gaussian 1.xls

3. Gaussian 2a.xls

4. Gaussian 2b.xls

5. Gaussian 3.xls

6. Gaussian 3b.xls

7. Gaussian 4.xls

8. Gaussian 4b.xls

9. Top Five Cancers.xls

10. US Income.xls

Test Results

1. Breast Cancer Wisconsin.mat

2. Gaussian 1.mat

3. Gaussian 2a.mat

4. Gaussian 2b.mat

5. Gaussian 3.mat

6. Gaussian 3b.mat

7. Gaussian 4.mat

8. Gaussian 4b.mat

9. Top Five Cancers.mat

10. US Income.mat

MATLAB® Source Code

1. bsearch.m – Performs a binary search for a number through a range of sorted numbers. The function returns the index of the first number matching the search number.

2. EvalTree.m – Evaluates a binary decision tree by measuring the accuracy, error rate, precision, recall, F-measure, node count, depth count, and complexity on a test data set.

3. EvalTreeGUI.fig – Contains a complete description of the EvalTreeGUI layout and the components of the GUI: push buttons, menus, axes, and so on.

4. EvalTreeGUI.m – Controls the EvalTreeGUI, including the callbacks for its components. For a complete reference on this application, please refer to Section 4.1.4.

5. FixCatsplit.m – Changes the stored names of categorical splits in a decision tree structure from a number to their actual text names. These modifications make constructed decision trees easier to understand in the Tree Viewer application.

6. GetDataFromXls.m – Loads data from an Excel spreadsheet into an Object structure.

7. IndexShuffle.m – Produces a randomly shuffled array of unique whole numbers from 1 to n.

8. num2cellstr.m – Converts all elements of a two-dimensional array of numbers into a two-dimensional array of cell strings.

9. obj2info.m – Converts an Object structure produced by GetDataFromXls into an information system representation.

10. removenans.m – Replaces all NaN values in a two-dimensional array with an empty value.

11. replacenans.m – Replaces all NaN values in a two-dimensional array with a user defined number.

12. replacevals.m – Replace all instances of a number in a two-dimensional array with another user defined number.

13. roughtree.m – Constructs a binary decision tree using the Rough Product splitting criterion.

14. stripvals.m – Produces a single-dimensional array of values consisting of the complement of the intersection of two arrays.

CHAPTER ONE

INTRODUCTION

Synopsis

This chapter begins with Section 1.1, which discusses induction for pattern recognition and some difficulties associated with it. Section 1.2 states our research objective. Section 1.3 lists the contributions from the work in this thesis.

1.1 General Problem

Pattern recognition is a natural and useful ability shared by all intelligent creatures. Stimuli from the environment trigger countless instinctual and learned responses in the brain, causing it to make decisions based on patterns in the stimuli. Using this deceptively simple process, we are able to recognize faces, identify music on the radio, avoid foul smells, massage tense muscles, or savor the satisfaction of eating our favorite foods – just to name a few. It is, therefore, not surprising that we would like to mimic this ability in the intelligent machines we build. Currently, various researchers are actively developing pattern recognition systems in areas such as machine vision [3, 4, 19, 20, 28, 36, 40, 42], automotive design optimization [1, 5], control systems [8, 27], and genetics [6, 50].

Pattern recognition highly correlates to logical reasoning, since both areas classify the structure of statements and arguments. Philosophers generally partition

logical reasoning into two major categories: inductive reasoning and deductive reasoning.

In inductive reasoning, we form generalizations based on observations. However, we cannot assume these generalizations as true for all cases unless we observe all cases. This presents a fundamental problem with induction: it assumes that an event in the future will always occur the same way as it has in the past. There is no justification for this assumption – it is simply accepted, but unexplained. The reason for this "faith" in induced models is that extreme skepticism is not practical for real-world pattern recognition. It is often simpler to understand the core concepts within a set of observations than to be able to explain every possible reason for why an observation occurred.

In deductive reasoning, we form conclusions about observations based on generalizations. Unlike inductive reasoning, we can formally prove a deduced conclusion as true if its premises are true. For example, let us establish the following premises: (a) all trees consist of wood and (b) we observe several instances of wood burning. If we assume these premises as true, then we can logically deduce that (c) some trees can burn. However, since we did not observe all instances of wood burning, we cannot claim that all trees can burn.

Deductive reasoning relies on generalizations that we usually produce with induction. Therefore, for the pattern recognition paradigm, our goal is induce the most representative models for any set of observations. Various researchers have proposed numerous supervised [4, 7, 16, 23, 33, 34, 35, 43, 44, 45, 46, 47, 48] and unsupervised [2, 9, 10, 11] induction algorithms to meet this goal. However, since there is no

generally accepted theory for induction, we can only verify induced models by experiment.

Much of the difficulty in inducing models revolves around vagueness within a set of observations [31]. In many real-world situations, we cannot precisely determine a concept because information is missing, noisy, or unequivocally wrong. Usually, we make assumptions about our observations (such as common sense reasoning) to simplify the induction in these cases. However, these assumptions may not be applicable in every case.

Rough set theory appears to provide a sound theoretical basis for inductive reasoning [31]. It can express vagueness by employing a boundary region around a concept. The size of the boundary region relative to that of the precise concept indicates the relative vagueness within a concept. An empty boundary region for a concept implies a precisely defined concept, whereas a non-empty boundary region implies an imprecisely defined concept due to insufficient knowledge. Zdzisław Pawlak, the pioneer of rough set theory, wrote the following about inductive reasoning and rough set theory:

> The creation of computers and their innovative applications essentially contributed to the rapid growth of interest in inductive reasoning. This domain develops very dynamically thanks to computer science. Machine learning, knowledge discovery, reasoning from data, expert systems and others are examples of new directions in inductive reasoning. It seems that rough set theory is very well suited as a theoretical basis for inductive reasoning. Basic concepts of this theory fit very well to represent and analyze knowledge acquired from examples, which can be next used as a starting point for generalization. Besides, in fact rough set theory has been successfully applied in many domains to find patterns in data (data mining) and acquire knowledge from examples (learning from examples). Thus, rough set theory seems to be another candidate as a mathematical foundation of inductive reasoning. [31]

Rough set theory provides researchers and data analysts with data mining techniques [13, 22] to characterize concepts using few assumptions. In essence, it lets the data speak for itself. Researchers have exploited rough set theory in applications such as attribute discretization [27], dimensionality reduction [39], knowledge discovery [16, 48], and time-series data analysis [15]. Because rough set theory demonstrates strong potential for induction, we choose it as the basis for our research. For more information about rough set theory, we recommend reading Section 3.1 in this thesis.

1.2 Research Objective

We aim to advance induction for pattern classification using rough set theory as our basis. We use the binary decision tree as our base model primarily for its simplicity and widely-understood properties. Because we restrict ourselves to univariate node splitting, this model allows us to cast a decision at any node as a one-dimensional optimization problem [12].

1.3 Contributions

Our research contributes to the overall knowledge base for induction algorithms in pattern classification. Specifically, our contributions are as follows:

- We developed the Rough Product – a rough set based measure of information.

- We developed a Rough Product based splitting criterion for binary decision tree construction.

- We developed the EvalTreeGUI software tool to empirically analyze the classification and structural metrics of the Gini Index, Twoing Rule, Maximum Deviance Reduction, and Rough Product splitting criteria.

- We conducted experiments on artificial and real-world data to comparatively analyze the Gini Index, Twoing Rule, Maximum Deviance Reduction, and Rough Product splitting criteria.

Conclusion

In this chapter, we discussed induction for pattern recognition and some associated difficulties due to vagueness in observations. We also stated our research objective and listed the contributions of this thesis.

CHAPTER TWO

DECISION TREE OVERVIEW

Synopsis

In this chapter, we provide a general overview for decision trees. Section 2.1 describes the decision tree structure and introduces terminology that we will use throughout the thesis. Section 2.2 discusses time complexity issues regarding the construction of optimal decision trees, and justifies the development of heuristics to create near-optimal decision trees. Section 2.3 discusses decision tree classifier design, and examines data preprocessing, splitting criteria, stopping criteria, and pruning methods.

2.1 Decision Tree Description and Terminology

A decision tree is a logical structure that is both descriptive and predictive, as shown in Figure 2.1. These qualities come from the notion that we can visually inspect a tree structure and understand how it maps observations about a collection of objects to a target value. A decision tree accepts as input an object described by a set of attributes and produces as output a single prediction. Predictions can be either for classification (i.e., to choose a distinct category to which an input belongs) or for regression (i.e., to calculate a real value). Our work deals exclusively with classification.

We now define some basic terminology that we will use throughout the thesis. A *tree* is a directed acyclic graph that satisfies the following properties [37]:

6
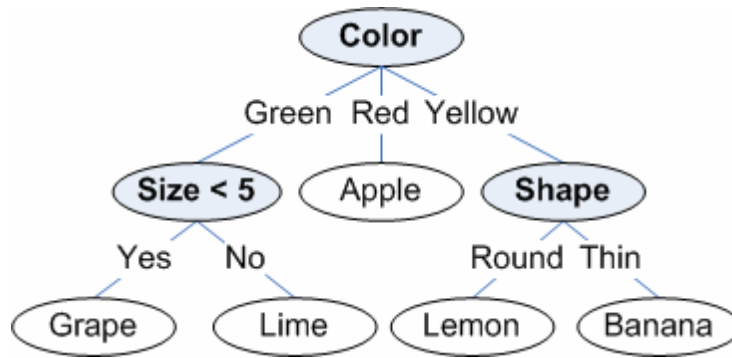
Figure 2.1: A Decision Tree Classifier.

1. There is exactly one node, known as the root, which no branches (or edges) enter.

2. Every node, except the root, has exactly one entering branch.

3. There is a unique path (or sequence of branches) from the root to each node in the tree.

If a branch exists from node *A* to node *B*, then we consider node *A* as a *parent* node and node *B* as a *child* node. We consider nodes without children as *leaf* nodes or *terminal* nodes; all other nodes are *internal* nodes. A *binary tree* is a special type of tree in which every node has either no children or exactly two children.

A tree becomes a decision tree when each internal node is a conditional separator and each terminal node is a prediction. A *split* is another term for a conditional separator at a node; it corresponds to split a collection of objects among a set of child nodes. For example, the root node in Figure 2.1 splits the collection of fruit objects based on color. A decision tree may contain univariate or multivariate splits. Univariate splits partition a collection of objects using a single attribute. Multivariate

splits partition a collection of objects using a combination of attributes. Our work

exclusively considers univariate splits.

2.2   Decision Tree Time Complexity

While a problem may be computationally solvable in principle, it may not be

solvable in practice due to an inordinate amount of time required to solve it. To

understand the time complexity of an arbitrary algorithm, we could measure the running

time by determining the maximum number of steps used by the algorithm on any input

of length n. However, since expressions for the exact running time of an algorithm are

often complicated, we usually estimate them through asymptotic analysis, which seeks

to understand an algorithms' running time on very large inputs.

Within the framework of Time Complexity Theory, we define two primary

classes of time complexity. The class P contains decision problems that can be solved

on a deterministic Turing machine that is polynomial in time with respect to the size of

the input. The class NP contains decision problems that can be verified as solved in

polynomial time with respect to the size of a given input (or equivalently, only decided

in polynomial time on a nondeterministic polynomial time Turing machine). Note that

the nondeterministic Turing machine does not correspond to any real-world computing

device, but rather is a useful mathematical definition that characterizes problems in NP.

Nondeterministic Turing machines are proven [41] to exhibit exponential time

complexity on equivalent deterministic Turing machines.

In general, polynomial differences in running time growth rates are considered

small, whereas exponential differences are considered large. For example, an input of

size 1000 (i.e., $n = 1000$) in the polynomial time algorithm n^2 produces one million

steps. On the contrary, the same input in the exponential time algorithm 2^n produces a

humongous number virtually beyond comprehension. This dramatic difference

indicates that polynomial time algorithms (i.e., in class P) tend to be practically solvable

on a computer, whereas exponential time algorithms (i.e., in class NP) tend to be

practically unsolvable.

Optimal decision tree construction (in the sense of minimizing the expected

number of tests required to classify an unknown sample) has been proven to be in the

class of NP-complete [17], which is a special class of problems in NP whose individual

complexity is related to that of the entire NP class. The importance of this result can be

measured by the amount of work that has been put into finding efficient algorithms for

optimal decision tree construction. It is conjectured that no such efficient algorithms

exist (on the belief that $P \neq NP$) and therefore, there is motivation for finding efficient

heuristics for constructing near-optimal decision trees [25, 32].

2.3   Decision Tree Classifier Design

Classifiers constructed through an inductive learning process attempt to

correctly map a data item (which is described by a collection of attributes) into one of

several predefined classes. The inductive learning process takes a set of data item

samples and their corresponding class labels as input, and creates a classification model

that predicts the class labels of unlabeled data items. This section focuses on the design

of a decision tree classifier and discusses some popular heuristics for preprocessing,

node splitting, split stopping, and tree pruning. We do not intend this section to be a

survey of decision tree construction techniques; for such detailed information, we refer

the reader to [26, 37]. In this section, we assume that raw data is available, and

therefore do not discuss input sensing, segmentation, and feature extraction as it relates

to a pattern recognition system.

2.3.1 Data Preprocessing

Typically, a raw data set contains unaltered or unprocessed samples from a

population. The most useful raw data accurately represents the population from which

it came. A combination of good statistical sampling and expert judgment can help to

ensure that this is true.

In the real world, raw data generally contains a certain degree of noise. This

could be due to a variety of reasons, such as missing data, incorrectly measured data,

and/or outliers (samples not from the population in question). As such, we often

introduce preprocessing (i.e., "cleaning") to minimize the effect of – or even completely

remove – the noise in raw data. This preprocessing phase may also transform our raw

data into a more useful representation, such as an information system, as shown in

Section 3.1.1. While many preprocessing steps are application-specific, we briefly

mention a few techniques that are problem-independent and may help to improve

classifier performance. More information on preprocessing may be found at [18].

2.3.1.1 Data Smoothing

Data smoothing is sometimes useful when a numeric attribute has many distinct

values, particularly if minor differences between values are insignificant and could

possibly degrade classifier accuracy and performance. We have various methods to

smooth values, which include averaging similar values, rounding values to the nearest whole number, or separating continuous value ranges into distinct classes.

2.3.1.2 Automatic Data Replacement

Automatic data replacement may be useful if the data is missing values. However, we should only use this technique if dropping incomplete samples is unfeasible. Data may be automatically generated by replacing all missing values with a single global constant, their attribute mean, or their attribute mean for a given class. Unfortunately, these techniques bias our data. To offset the bias, it may be possible to develop a predictive model that guesses the most likely value by using the most information in the data. The main flaw with this approach, however, is that the predicted value is not the correct value. Fortunately, many tree-growing algorithms account for missing values by assuming that samples with missing values are distributed probabilistically according to the relative frequency of known values.

2.3.1.3 Automatic Outlier Detection

Automatic outlier detection is useful for discovering samples that are inconsistent with the overall data. These inconsistencies might be due measurement error, or perhaps may truly represent the data's inherent variability. In either case, when we discover an inconsistency, we must make a decision on how to handle it. Typically, we prefer to minimize its influences.

There are several approaches to outlier detection. The simplest approach assumes a particular distribution for an attribute and detects the samples that are beyond a defined distribution threshold. Another approach, which we can apply to multiple

11

numeric dimensions, uses distance calculations to detect samples that are farthest from the majority of samples. A third (and more complex) approach defines basic characteristics about a data set and detects samples that deviate from them.

2.3.2   Splitting Criteria

The splitting criterion is the core for all tree-building algorithms. The basic principle underlying these criterions is that of simplicity; that is, we prefer splitting on attributes that will ultimately lead to a simple, compact classifier [12]. Normally, the splitting criterion tries to make each child node as pure and relevant as possible. However, the vast major of splitting criterions are greedy heuristics performing successive local optimizations, which do not guarantee a global optimum. There is also no guarantee that the terminating nodes will classify to a single decision class. Nevertheless, decision tree classifiers constructed in this manor have been used successfully in real world situations [7, 24, 37, 51].

The CART (Classification and Regression Tree) algorithm [7] typically uses the Gini Index criterion. We can think of the Gini Index as a node's expected error rate when a class is selected randomly from that node's class distribution [12]. As an impurity measure, it reaches its minimum value (i.e., zero) when a node contains only one class. It reaches its maximum value when the frequency for every class in a node is the same.

Let function f be the frequency of class j at node N. The Gini Index is calculated as follows:

$$I_G(N) = 1 - \sum_j f(N,j) = \sum_{j \neq k} f(N,j)f(N,k).$$

The Maximum Deviance Reduction splitting criterion is similar to the Gini Index, but is more commonly recognized as the Entropy function:

$$I_E(N) = -K \sum f(N, j) \log f(N, j).$$

This criterion is most frequently used in the ID3 [33], C4.5 [35], and C5.0 tree growing algorithms. It originates in information theory [38], where entropy is used to measure the disorder in a signal or random event. Like in the Gini Index, the higher the entropy (or uncertainty), the more information is required to completely describe the data.

The goal of the Gini Index and Maximum Deviance Reduction is to reduce the uncertainty until a pure leaf node is established. Therefore, the best heuristics choose a split that most decreases the uncertainty. For binary decision trees, this heuristic is defined as the change in information:

$$\Delta I(N) = I(N) - P_L I(N_L) - (1 - P_L) I(N_R),$$

where N_L and N_R are the left and right child nodes, and P_L is the fraction of samples in the parent node N that go to N_L.

The Twoing Rule splitting criterion has a much different splitting strategy than Gini Index or Maximum Deviance Reduction, and tends to be most useful in multi-class binary tree creation [12]. It chooses the split that best partitions *groups* of classes in a node. The algorithm creates a candidate "super-group" G_1 that consists of all samples in some subset of classes, and another candidate "super-group" G_2 that consists of the remaining samples. Then the algorithm searches for a split that best separates the two 'super groups' by computing the change in information (as if it was a standard two-class problem) and choosing the split that maximizes the change in information.

13

2.3.3 <u>Stopping Criteria</u>

An important part of constructing a decision tree classifier is deciding when it is

no longer useful to split a node. Consider the case where we grow a tree to its lowest

impurity; typically, such trees over fit the data and have poor generalization.

Alternatively, consider the case where we prematurely stop tree growth; such trees

under fit the data and typically have high error rates. The stopping criterion detects

when a node sufficiently describes a portion of the data and designates it as a

terminating node.

We cover several stopping criteria commonly used in decision tree construction.

Traditional stopping criteria use validation sets or cross validation, as described in

Section 4.1.1. They test the decision tree during construction with validation sets and

stop the algorithm when the validation set error is minimal. Other methods use preset

threshold values to stop a node split. They may place thresholds on criteria such as the

reduction in impurity, number of samples in a node, percentage of samples in a node, or

tree depth. A significant drawback with these methods is that it is usually difficult to

know how to preset a threshold since a simple relationship between a threshold and

classifier performance rarely exists. The minimum description length (MDL) method

[34] attempts to trade complexity for accuracy by splitting until it reaches a minimum in

the global criterion function

$$\alpha \cdot size + \sum_{\text{leaf nodes}} I(N),$$

where $size$ is a measure of tree complexity, α is some positive constant, and $I(N)$ is

the impurity based on entropy. The MDL criterion shows us that the larger the tree, the

14

lower the terminal node entropy; its goal is to find an optimal balance between the tree

size and terminal node entropy. However, a drawback to implementing the MDL

criterion is that a simple relationship between α and classifier performance is difficult

to find. We may also use the statistical significance of the impurity reduction as a

stopping criterion. Essentially, we determine if an estimated distribution of ΔI is

statistically different from zero, for instance, by a chi-square test. If the change is

significant, we continue to split; otherwise, we stop splitting.

2.3.4   Pruning Methods

Pruning is an alternative to the stopping criterion. Rather than stopping the node

splitting during construction, we fully grow the tree to its maximum depth and then

eliminate pairs of terminal nodes whose removal yields a satisfactory small increase in

impurity. As such, parent nodes of eliminated children become new terminal nodes.

Other forms of pruning replace complex sub-trees with terminal nodes directly. In

addition, certain pruning methods employ cross validation, bootstrap, and MDL

procedures. Generally, we may prune with any stopping criteria used during node

splitting – only in reverse. [14] provides an excellent survey of various pruning

methods and an empirical analysis of pruning on both predictive accuracy and size of

induced decision trees.

Pruning is generally preferred over stopping criteria because it does not suffer

from a lack of sufficient look ahead, known as the horizon effect [12]. Furthermore,

cross validation and bootstrapping require that we partition the data into training and

testing sets. With pruning, we simply use all the data and prune unnecessary terminal nodes.

Conclusion

This chapter provided a general overview for decision trees. We described decision trees and introduced important terminology. In addition, we discussed time complexity issues for optimal decision tree construction and justified the development of heuristics to create near-optimal decision trees. We also discussed decision tree classifier design, examining data preprocessing, splitting criteria, stopping criteria, and pruning methods.

CHAPTER THREE

ROUGH SET BASED DECISION TREE CONSTRUCTION

Synopsis

The purpose of our research is to formulate a measure of information based on rough set theory that we apply in a splitting criterion for a binary decision tree classifier. In Section 3.1, we introduce the concepts of rough set theory and show how we characterize unknown decision boundaries by defining lower and upper approximations. In Section 3.2, we examine past research related to rough set-based feature selection. In Section 3.3, we derive and apply the rough product measure of information. Section 3.3.1 mathematically develops the rough product information measure. Section 3.3.2 performs a sample rough product calculation on a single conditional attribute and analyzes the results. Section 3.3.3 presents an algorithm that applies the rough product in a splitting criterion for a binary decision tree classifier.

3.1 Rough Set Theory

Zdzisław Pawlak first introduced rough set theory in 1982 as way to deal with vague or uncertain data [29]. It is an extension to classical set theory used to approximate definite and possible sets from empirical data. Researchers have applied the theory in various branches of artificial intelligence, such as machine learning, expert systems, and pattern recognition [22]. The theory provides a means to reduce a data set, evaluate significant features in data, offer clear interpretations for results, and identify

17

relationships normally not found using statistical methods [30]. In addition to rough set

theory, this section introduces two important probabilistic measures used to analyze

decision rules: the *certainty factor* and the *coverage factor*.

3.1.1   Decision Tables

We typically represent empirical data in a two-dimensional matrix as shown in

Table 3.1. The rows of the matrix represent individual sample objects and the columns

represent sample attributes.

Mathematically, we define the empirical data as an information function that

consists of the cross product between all sample objects and all attributes, yielding an

attribute value

$$\rho : U \times A \to V \,,$$

where U is the set of all objects, A is the set of all attributes, and V is the set of all

attribute values.

Table 3.1

Empirical Data Example

	a_1	a_2	a_3
u_1	v_1	v_3	v_6
u_2	v_1	v_4	v_7
u_3	v_2	v_5	v_7

For Table 3.1, $U = \{u_1, u_2, u_3\}$, $A = \{a_1, a_2, a_3\}$, and

$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. If we define ρ as the information function for Table 3.1,

then $\rho(u_1, a_1) = v_1$, $\rho(u_1, a_2) = v_3$, $\rho(u_1, a_3) = v_6$, and so on.

To represent our empirical data as a decision table, we must partition the set of

all attributes A into two mutually exclusive subsets: the subset of conditional attributes

C and the subset of decision attributes D. For this thesis, we assume that the decision

attribute subset contains only one element designated for classification; however, in the

general case, D may contain any number of elements.

Generally, attribute values within a decision table can be either categorical or

numeric. To use rough set theory, however, we must ensure that attribute values within

a decision table are represented categorically. This normally requires us to bind the

numeric attributes with defined numeric value ranges. Values within the lower and

upper bounds of a range become mapped to a distinct category defined for that range of

values.

3.1.2 <u>Indiscernibility</u>

The indiscernibility concept is central to rough set theory. Essentially, it is a

binary equivalence relation for a pair of objects within a decision table. Two objects

are considered indiscernible if one cannot distinguish between the objects based on a

given set of attributes. If B is a non-empty subset of A, then the indiscernibility relation

$IND(B)$ is a relation on U defined for $x, y \in U$ such that $(x, y) \in IND(B)$ if and only if

$\rho(x, a) = \rho(y, a)$ for all $a \in B$.

The indiscernibility relation can essentially partition U into equivalence classes, denoted by $U / IND(B)$, which are collections of indiscernible objects. For example, in Table 3.1, the partitions of equivalence classes with respect to A_1 can be shown as $U / IND(a_1) = \{\{u_1, u_2\}, \{u_3\}\}$. In this case, we see that u_1 and u_2 are indiscernible. However, if we consider $U / IND(a_1 \cup a_2)$, then u_1 and u_2 are not indiscernible because $U / IND(a_1 \cup a_2) = \{\{u_1\}, \{u_2\}, \{u_3\}\}$.

3.1.3 Rough Sets

Using equivalence classes, we can form sets of objects that produce the same outcomes in the decision attributes. However, objects contained within the same equivalence class will often have different outcomes. We may handle this type of impreciseness with rough set theory through set approximations. By evaluating the certainty of an outcome occurring within an equivalence class, we can classify subsets of objects into lower, upper, and boundary approximations.

The lower approximation of a set with respect to the data set is the collection of objects whose equivalence classes are fully contained within the set of objects we want to approximate. Therefore, if we want to approximate a concept given by a set of objects $X \subseteq U$ using a subset of conditional attributes $R \subseteq A$, then the lower approximation for R is

$$R_*(X) = \bigcup \{E \in U / IND(R) \mid E \subseteq X\}.$$

The upper approximation of a set with respect to the data set is the collection of objects whose equivalence classes are possibly contained within the set of objects we

want to approximate. We note that the upper approximation always contains the lower

approximation. So again, if we want to approximate a concept given by a set of objects

$X \subseteq U$ using a subset of conditional attributes $R \subseteq A$, then the upper approximation

for R is

$$R^*(X) = \bigcup \{E \in U / IND(R) \mid E \cap X \neq \varnothing\}.$$

The boundary approximation of a set with respect to the data set is the collection

of objects whose equivalence classes cannot be classified as either contained or not

contained within the set of objects we want to approximate. This is equivalent to

finding the set difference between the upper and lower approximations:

$$BND_R(X) = R^*(X) - R_*(X).$$

We can define a rough set using these approximations, as illustrated in

Figure 3.1. Simply, a set is rough if its boundary condition is not empty; otherwise, the

set is definable based on R.

The positive region of decision classes is a collection of objects in U that we can

classify with complete certainty to $U / IND(D)$ employing the attributes of C:

$$POS_C(D) = \bigcup R_*(X).$$

3.1.4   Decision Rules

A decision rule is an implication statement in the form $\Phi \rightarrow \Psi$, where Φ are

the conditions connected with the AND operator and Ψ are the decisions connected

with the OR operator. We can directly convert rough sets into decision rules by relating

21
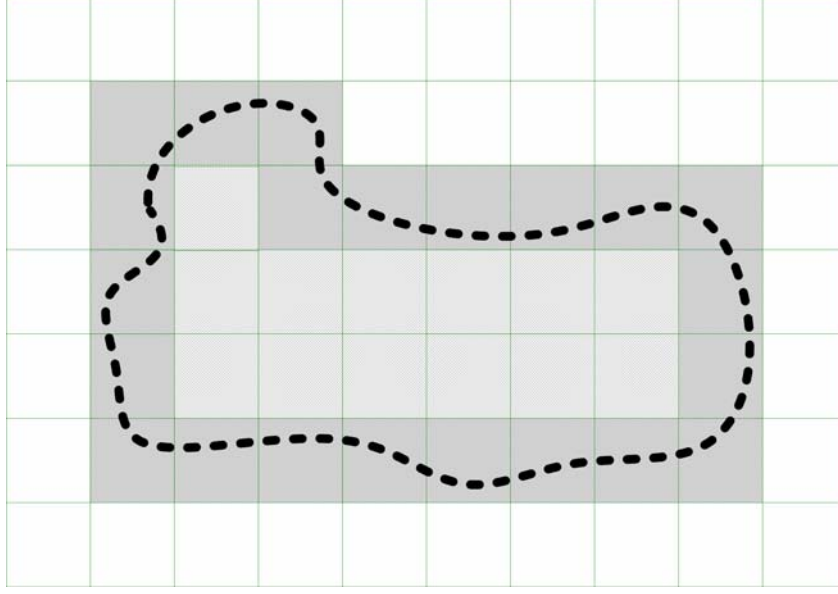
Figure 3.1: Visualization of a Rough Set. Each box in the grid represents an equivalence class. The dashed black line represents the boundary of the actual, but unknown set. The lightly shaded boxes within the boundary line represent the lower approximation. The heavily shaded boxes represent the boundary approximation – or rough set. The union of the lightly shaded and heavily shaded boxes represents the upper approximation.

the conditional attributes from its equivalence classes to its decisions. Typically, before

we transform rough sets into decision rules, we reduce the rough sets to remove

superfluous data while completely preserving the consistency of the original data. That

is, $R \subseteq C$ is said to be a *D*-reduct when $POS_R(D) = POS_C(D)$ and there is no $R' \subset R$

such that $POS_{R'}(D) = POS_C(D)$. Algorithms for data reduction are outside the scope

of this section and thus are not covered. However, to transform rough sets into decision

rules, data reduction is not required, and in some cases not always possible in a

reasonable time period due to an extremely large data set.

The probabilistic properties of certainty and coverage may be applied to these newly formed decision rules to extract previously unknown information.

The certainty factor is defined as

$$P(\Psi \mid \Phi) = \frac{\text{number of cases satisfying } \Phi \text{ and } \Psi}{\text{number of all cases satisfying } \Phi}.$$

The coverage factor is defined as

$$P(\Phi \mid \Psi) = \frac{\text{number of cases satisfying } \Phi \text{ and } \Psi}{\text{number of all cases satisfying } \Psi}.$$

We note that all conclusions drawn from the application of rough set theory on a data set are only true with respect to that data set. We cannot assume that these conclusions are generalizations of the population from which the data set was collected, unless the data set is both sufficiently large and representative of the actual population.

3.2 Past Research with Rough Set Based Feature Selection

Here, we briefly survey several approaches to feature selection that have used rough set theory. This section does not intend to serve as an exhaustive survey of all rough set-based approaches, but rather familiarize the reader with the type of work that done in this area. The surveyed approaches include the PRESET algorithm, the ACRs algorithm, Significance Oriented Methods, and Support Heuristics.

3.2.1 PRESET Algorithm

The PRESET algorithm [23] is a heuristic feature selector that assumes a noise-free binary domain and uses rough set theory to generate an optimal sequence of attributes for object classification. This algorithm can construct a decision tree with a

23

very specific property: that all nodes at a given tree depth are split using the same attribute. However, PRESET is likely to fail on data whose attributes are highly correlated, since combinations of these attributes do not help in finding relevant splits [49].

3.2.2   ACRs Algorithm

The ACRs algorithm [44] splits on a node by finding a single conditional attribute that minimizes the size of the rough set corresponding to this attribute. It first calculates the lower approximation, upper approximation, negative region, and boundary region for all attributes in a data set. Then, the algorithms chooses the attribute with the smallest boundary as the split that separates the samples in the node along all attribute values. Because the split uses all attribute values at the node, the algorithm does not need to analyze the chosen attribute in deeper nodes, which improves the calculation speed. If all samples inside a node belong to the same class, the node becomes a leaf; otherwise, the splitting continues. Issues such as the treatment of a split when multiple attributes have the same minimum rough set size are discussed in [43].

3.2.3   Significance-Oriented Method

The significance-oriented method [16] uses rough set theory to find the most significant conditional attribute for splitting. The significance of an attribute is defined as the increase in dependency between conditional attributes and the decision attribute that results from the addition of a single attribute. The method selects the attribute

whose dependency increases the quickest. The dependency between the conditional

and decision attributes is defined as

$$g(R,D) = \frac{card\big(POS_R(D)\big)}{card(U)}.$$

The significance of an attribute a is defined as

$$SIG(a) = g(R+a,D) - g(R,D)$$

where R is the set of currently selected features and D is the decision attribute.

3.2.4 Support-Based Heuristics

The maximum support heuristic [46, 47] was developed to address a

shortcoming of the significance oriented method. The significance-oriented method

only considers the dependency of an additional feature, but fails to consider the number

of instances covered by the rules generated due to the additional feature. This

additional information is known as the "support" of the rule.

The purpose of the maximum support heuristic is to search for the most

significant rule with the largest support. Rules with high significance but low support

are inherently not useful for classification. For example, in an extreme case, a feature

with a unique value for each sample in a data set can trivially be used to generate rules

for classification of any other feature in that data set. This feature certainly has very

high dependency. However, these rules would highly over fit the data because the

maximum number of instances for each rule is one.

The maximum support heuristic is defined as:

$$F(R,a) = card\big(POS_{R+\{a\}}(D)\big) \times MAXSize\big(POS_{R+\{a\}}(D) / IND(R+\{a\})\big).$$

25

The first factor indicates the size of the consistent instances. The second factor denotes the maximum size from the indiscernibility classes included in the positive region (i.e., the support).

The maximum support heuristic is limited because it selects the attributes that produce only a single rule with high significance and support [46]. Consequently, other forms of the support heuristic were developed. The average support heuristic, for example, was created to take into account the overall quality of the most significant rules generated by an additional attribute. The overall quality of the n most significant rules is:

$$Q(R,a) = \frac{1}{n}\sum_{i=1}^{n} S(R,a,d_i)$$

where $S(R,a,d_i) = MAXSize\left(POS_{R+\{a\}}(D = d_i) / IND(R + \{a\})\right)$ is the support of the most significant rule for decision class $\{D = d_i\}$. The domain of D is $\{d_1, d_2, ..., d_n\}$.

Thus, the average support heuristic is the product between the size of the consistent instances and the overall quality.

$$F(R,a) = card\left(POS_{R+\{a\}}(D)\right) \times Q(R,a)$$

The average support heuristic was further expanded to the parameterized average support heuristic (PASH) [45, 46], which relaxes the traditional notion of the lower approximation for a rough set. In PASH, the lower approximation is allowed to hold a certain degree of impurity to better model data that has inconsistent instances. This extension of the lower approximation is based on the prior probabilities of the

26

decision classes. If the prior probabilities are known, then the lower approximation is defined as:

$$R_*(D = d_i) = \bigcup \left\{ \begin{array}{l} E_j \in U / IND(R) \mid \\ \dfrac{P(D = d_i \mid E_j)}{P(D = d_i)} = MAX \left\{ \dfrac{P(D = d_i \mid E_j)}{P(D = d_k)}, 1 \le k \le n \right\} \wedge \\ P(D = d_i \mid E_j) > t(t \ge 0.5) \end{array} \right\}$$

If the prior probabilities are unknown, then we assume each value of a decision attribute has equal prior probability. In this case, the lower approximation is defined as:

$$R_*(D = d_i) = \bigcup \left\{ E_j \in U / IND(R) \mid P(D = d_i \mid E_j) > t(t \ge 0.5) \right\}$$

The advantage of PASH is that it considers the overall quality of the rules while including predictive rules that are ignored using traditional rough set theory.

3.3 The Rough Product

The average support heuristic demonstrates that we can remove all noise in the data set by taking into account the average support of the most significant rules for every decision class. Essentially, this heuristic only determines the most significant rules found in the lower approximation. PASH improves the accuracy of the average support heuristic by assuming that measured data will always have a certain degree of noise. It includes certain rules that most likely describe the lower approximation according to a predefined probability threshold. However, by including these additional rules, PASH increases the overall complexity of the rule-based classifier.

Generally, the best classifiers optimally balance two key metrics: accuracy and complexity. These metrics are directly affected by the amount of noise tolerated during

training. The research described in Section 3.2 shows that rough set theory could be used to discriminate between pure and impure data by using lower and upper approximations. Therefore, our research effort focuses on two parts:

1. The creation of a new measure of information based on rough set theory.

2. The development of a splitting criterion for a binary decision tree classifier using our new measure of information.

3.3.1   Derivation of the Rough Product

Let us define an information function such that

$$\rho : U \times A \rightarrow V ,$$

where U is the set of all objects, A is the set of all attributes, and V is the set of all attribute values. We then partition the attributes A into two subsets C and D, respectively known as the condition and decision attributes. Next, we define the set of values for an attribute $a \in A$:

- For the categorical case, $V^a = \{v \mid \forall x \in U : v \in \rho(x,a)\}.$

- For the numeric, non-categorical case with a split threshold λ,

$$V^a = \{v \mid v \in \{true,\ false\} \wedge \forall x \in U \wedge \lambda \in \Re : v = (\rho(x,a) < \lambda)\}.$$

Using V^a, we define the set of unique values for an attribute $a \in A$ such that

$$UV^a = \{v \mid v \in (V^a \cap V^a)\}.$$

We define a sequence as a mapping of natural numbers in N_1 (set of positive whole numbers greater than or equal to 1) to elements in a set. If we let X be an arbitrary set, then a sequence of X is defined as:

28

$$\langle X \rangle \overset{def}{=} \{ f : N_1 \to X \mid \operatorname{dom} f = 1,2,...,card(X) \}.$$

Using the sequence of unique values for an attribute $\langle UV^a \rangle$, we construct an

indiscernibility matrix between the values of attribute $c \in C$ and the values of attribute

D. The arrangement of the matrix is critical for subsequent calculations. If we let

$m = card(UV^c)$ and $n = card(UV^D)$, we can define the dependency matrix M^c as:

$$M_{ij}^c = \begin{cases} p \in U / IND(c \cup D) \mid \\ c \in C \wedge \forall x \in p : (i, \rho(x,c)) \in \langle UV^c \rangle \wedge (j, \rho(x,D)) \in \langle UV^D \rangle, \\ \text{for } i = 1,2,...m \text{ and } j = 1,2,...n \end{cases}.$$

At this point, we no longer need to work with a matrix of sets, since each object

in each set is indiscernible to the other objects in the same set. As such, we can

collapse each set in M^c to a single number representing its cardinality and form a new

matrix K^c:

$$K_{ij}^c = \begin{cases} card(k) \mid c \in C \wedge k \in M_{ij}^c, \\ \text{for } i = 1,2,...m \text{ and } j = 1,2,...n \end{cases}.$$

From K^c, we determine the following results:

- The total number of objects with the same value in UV^c

$$CO_i^c = \sum_{j=1}^{n} K_{ij}^c \text{ for } i = 1, 2, ..., m.$$

- The total number of objects with the same value in UV^D

$$DO_j^c = \sum_{i=1}^{m} K_{ij}^c \ \text{ for } j = 1, 2, \ldots, n.$$

- The total number of all objects

$$TO^c = \sum_{i=1}^{m} \sum_{j=1}^{n} K_{ij}^c .$$

- The certainty factor for each indiscernible conditional value and decision value pair

$$CERT_{ij}^c = \frac{K_{ij}^c}{CO_i^c} \ \text{ for } i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.$$

- The coverage factor for each indiscernible conditional value and decision value pair

$$COV_{ij}^c = \frac{K_{ij}^c}{DO_j^c} \ \text{ for } i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.$$

- The certainty factor for each decision value independent of any conditional value

$$CERT_j^D = \frac{DO_j^c}{TO^c} \ \text{ for } j = 1, 2, \ldots, n.$$

One of the goals of our measure of information is to understand the manner in which the upper approximation for each decision value in UV^D changes when the set of objects is partitioned by a conditional value in UV^c. We do this by taking the difference between the certainty dependency matrix and the certainty independent of any conditional value. We call this the *rough gain*:

$$RG_{ij}^c = CERT_{ij}^c - CERT_j^D \text{ for i = 1, 2, ..., } m; \text{ j = 1, 2, ..., } n.$$

A positive rough gain indicates an improvement in the upper approximation, whereas a negative rough gain indicates its degradation. However, we cannot solely examine the rough gain to determine the preferred partition relative to the available objects because we could produce a partition with an extremely high positive rough gain for only an extremely small proportion of objects. Therefore, we must also examine the proportion of objects affected by the partition – in other words, the coverage of the condition and decision value dependency. Thus, we can define the rough set based measure of information, termed as the *rough product*, as follows:

$$RP_{ij}^c = RG_{ij}^c \times COV_{ij}^c \text{ for i = 1, 2, ..., } m; \text{ j = 1, 2, ..., } n$$

As in the rough gain matrix, positive and negative values in RP^c indicate an improvement or degradation in the upper approximation, respectively. However, by incorporating the coverage factors, the rough gain is scaled appropriately to reflect the support for each decision class. The highest positive values in RP^c for all *c* indicates attribute value partitions that produce the strongest increase in certainty with a high degree of support for the associated decision classes. Positive and negative values near zero indicate attribute value partitions that produce minimal effects on the associated decision classes, possibly suggesting noise. Values at zero indicate an attribute value partition that produces no effect on the associated decision classes. By using the rough product, we can understand the manner in which an attribute value partition affects the upper approximation for each decision class.

31

3.3.2 Example Rough Product Calculation

In this Section, we demonstrate a calculation for the rough product matrix. Our example uses Table 3.2 as an information function $\rho(x,a)$ to calculate the rough product matrix for the c_1 attribute.

First, let us define the set of all objects U, the set of all attributes A, and the set of all attribute values V for Table 3.2.

$$U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$

$$A = \{c_1, c_2, D\}$$

$$V = \{a, b, c, d_1, d_2, d_3\}$$

Next, we define the set of unique attribute values for attributes c_1 and D.

$$UV^{c_1} = \{a, b, c\}$$

$$UV^D = \{d_1, d_2, d_3\}$$

Now, we construct an indiscernibility matrix M^{c_1} between the values in UV^{c_1} and UV^D as shown in Table 3.3.

Using M^{c_1}, we construct the K^{c_1} matrix as follows:

$$K^{c_1} = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$$

Table 3.2

Demonstration Decision Table

	c_1	c_2	D
x_1	a	c	d_2
x_2	a	c	d_2
x_3	c	a	d_3
x_4	c	c	d_1
x_5	b	a	d_3
x_6	b	b	d_3
x_7	b	c	d_2

Table 3.3

Indiscerniblity Matrix for Attribute c_1

	d_1	d_2	d_3
a	$\varnothing$	$\{x_1, x_2\}$	$\varnothing$
b	$\varnothing$	$\{x_7\}$	$\{x_5, x_6\}$
c	$\{x_4\}$	$\varnothing$	$\{x_3\}$

From K^{c1}, we determine the following results:

$$CO^{c1} = \begin{bmatrix} 2 & 3 & 2 \end{bmatrix}^{\mathrm{T}}$$

$$DO^{c1} = \begin{bmatrix} 1 & 3 & 3 \end{bmatrix}$$

$$TO^{c1} = 7$$

$$CERT^{c1} = \begin{bmatrix} 0.0 & 1.0 & 0.0 \\ 0.0 & 0.\overline{33} & 0.\overline{66} \\ 0.5 & 0.0 & 0.5 \end{bmatrix}$$

$$COV^{c1} = \begin{bmatrix} 0.0 & 0.\overline{66} & 0.0 \\ 0.0 & 0.\overline{33} & 0.\overline{66} \\ 1.0 & 0.0 & 0.\overline{33} \end{bmatrix}$$

$$CERT^{D} = \begin{bmatrix} 0.14 & 0.43 & 0.43 \end{bmatrix}$$

Then, we calculate the rough gain and rough product as follows:

$$RG^{c1} = \begin{bmatrix} -0.14 & 0.57 & -0.43 \\ -0.14 & -0.1 & 0.23 \\ 0.36 & -0.43 & 0.07 \end{bmatrix}$$

$$RP^{c1} = \begin{bmatrix} 0.0 & 0.3762 & 0.0 \\ 0.0 & -0.033 & 0.1518 \\ 0.36 & 0.0 & 0.0231 \end{bmatrix}$$

By analyzing the rough product, we gain the following insight about the possible splits in the c_1 attribute:

- If we choose to split on value a, decision class d_2 yields to its lower approximation with relatively high relevance. We know this by noting the fairly high value in $M_{1,2}^{c1}$ and zeros in $M_{1,1}^{c1}$ and $M_{1,3}^{c1}$.

- If we choose to split on value b, the upper approximation for decision class d_3 will improve with some relative relevance. However, this improvement carries over a small amount of noise in class d_2.

- If we choose to split on value c, the upper approximation for decision class d_1 will significantly improve. However, this improvement also carries over a small amount of noise.

3.3.3 Rough Product Splitting Criterion for the Binary Decision Tree

This section documents an application of the rough product measure of information. We use the rough product in a splitting criterion for the binary decision tree model.

The general goal of any binary decision tree splitting criterion is to find not only the best feature upon which to split, but also the best value. In our splitting criterion, we define the "best" split as that which maximizes the sum of the maximum rough product in the left and right node, such that the decision classes for the maximum rough products in each node are not the same. In other words, we choose the split that best improves the upper approximation in each child node for two different classes.

We now describe the rough product splitting algorithm using pseudo-code. The input is the training data with designations for those attributes that have unordered categorical conditional values, ordered numeric conditional values, and unordered categorical decision values. Additionally, we set the *splitmin* for the tree, which is a number n such that impure nodes must have n or more objects to be split. The algorithm outputs a data structure that describes the decision tree. Since this algorithm

produces a classifier, at least one attribute in the data set must be categorical. For

brevity, we do not include initialization and resolution pseudo-code; we only focus on

the node-splitting portion of the algorithm.

Rough Product Splitting Algorithm

```
WHILE there are nodes available for splitting

    IF Node is impure with an object count >= splitmin THEN

        FOR each conditional attribute c

            IF c cannot be split THEN

                CONTINUE to next available attribute.

            ENDIF

            COMPUTE rough product RP_L for all left node splits.

            COMPUTE rough product RP_R for all right node splits.

            FOR each split s

                STORE the sum of the maximum rough products in RP_L
                and RP_R for split s, such that the decision classes
                associated with the maximum rough products in RP_L
                and RP_R for split s are not the same.

            ENDFOR

            STORE the maximum rough product sum for attribute c.

            STORE split associated with maximum rough product sum
            for attribute c.

        ENDFOR

        IF Node can be split THEN

            Split Node using the attribute and attribute value with
            the overall maximum rough product sum.

        ENDIF

    ENDIF

ENDWHILE
```

Conclusion

In this chapter, we developed a measure of information based on rough set theory. We defined the rough product for an attribute value partition as the product of the certainty factor gain and coverage factor. Using this measure, we analyzed the effect on each decision class caused by an attribute value split. The highest positive values in a rough product matrix indicate a strong increase in certainty with a high degree of support in the associated decision classes. Positive and negative values near zero indicate minimal effects on associated decision classes, which, in some cases, could be noise. Values at zero indicate a split that has no effect on associated decision classes.

We then applied the rough product measure of information in a splitting criterion for a binary decision tree. The splitting criterion prefers the split that maximizes the sum of the maximum rough product in the left and right node, such that the decision classes for the maximum rough products in each node are not the same. Other splitting criterions based on the rough product may be possible. In addition, it may be possible to apply the rough product in information-based algorithms.

CHAPTER FOUR

SPLITTING CRITERION ANALYSIS THROUGH CROSS VALIDATION

Synopsis

Our goal in this chapter is to comprehensively evaluate the Rough Product

splitting criterion by comparing its behavior to that of other similar splitting criteria. In

Section 4.1, we describe our approach for evaluating the Rough Product splitting

criterion. Section 4.1.1 explains our use of the k-folds cross validation method. Section

4.1.2 describes the metrics for measuring splitting criterion performance. Section 4.1.3

states the competing splitting criteria that we compare against the Rough Product

splitting criterion. Section 4.1.4 describes the EvalTreeGUI software tool that we use to

perform splitting criterion evaluations. In Section 4.2, we evaluate the Rough Product

splitting criterion using various data sets. Section 4.2.1 discusses the performance of

this criterion on artificially generated Gaussian distributed clusters. Section 4.2.2

discusses the performance of this criterion on real-world data.

4.1   Splitting Criterion Comparison Approach

According to [12], there are no context-independent or usage-independent

reasons to favor one classification method over another. When one algorithm

outperforms another, this occurrence is most likely a consequence of the algorithms' fit

to a particular pattern recognition problem. That said, we still wish to draw general

conclusions about the behavior of the Rough Product splitting criterion given arbitrary

data sets for classification. Thus, our purpose is two-fold: first, we must determine whether our splitting criterion is useful in classification; second, we must compare our resulting decision tree classifiers to other similar classifiers using various classification and structural metrics.

4.1.1 Methodology

Our analysis employs the k-folds cross validation technique. This technique takes an arbitrary data set and randomly divides its samples into k equally-sized disjoint subsets. A classifier is then trained k times, each time with a different subset held out for testing. The results from each testing subset are then combined – typically, by averaging – to estimate a classifier's performance. Estimation variances usually decrease as the value of k increases; however, k values equal to five or ten are generally considered reasonable. More information about cross validation for accuracy estimation and model selection is found in [21].

Because the k-folds cross validation technique is a heuristic, generalizations about a classifier's performance cannot be made without assumptions about the classifier or problem [12]. In our case, because each tested classifier is a binary decision tree, we assume performance differences are only due to splitting criteria. In addition, we implicitly make our assumptions about each evaluation data set without explicitly stating its prior information, distribution, and training data size.

4.1.2 Metrics

We determine classification metrics by comparing the classifier output to the true output for a given input. However, this comparison is not as simple as determining

whether the classifier output matches the true output. When a classifier predicts that a particular output class is true, it also predicts that all remaining output classes are false. Therefore, we must analyze all possible output classes to evaluate a prediction. For a given prediction and truth-value, each possible output class is designated as one of the following:

- A class is *true positive* when the classifier output correctly predicts that the class is equal to the true output.

- A class is *true negative* when the classifier output correctly predicts that the class is not equal to the true output.

- A class is *false positive* when the classifier output incorrectly predicts that a class is equal to the true output.

- A class is *false negative* when the classifier output incorrectly predicts that a class is not equal to the true output.

We illustrate the previous metrics with two cases shown in Table 4.1: (1) when the true output is equal to the classifier output and (2) when the true output is not equal the classifier output. We assume the set of output classes is {*A*, *B*, *C*}.

Table 4.1

Two Cases for Classifier Output Comparison

Situation	True Output	Classifier Output
Case 1	*A*	*A*
Case 2	*A*	*B*

In Case 1 (see Table 4.2), the classifier correctly predicts that the true output is class *A*. Thus, class *A* is true positive. The classifier also correctly predicts that the true output is neither class *B* nor *C*. Therefore, classes *B* and *C* are true negative.

In Case 2 (see Table 4.3), the classifier incorrectly predicts that the true output is class *B*, generating a false positive. The classifier should have predicted that class *A* is the true output. Since it did not, class *A* is false negative. Despite incorrectly predicting the true output, the classifier output still correctly predicts that class *C* is not the true output. Therefore, class *C* is true negative.

If we determine the total number of the true positives, true negatives, false positives, and false negatives for all test cases, then we can use a variety of performance metrics to analyze both the predictive performance of each output class and the predictive performance of the classifier as a whole. The following lists the classification performance metrics we use in our analysis:

Table 4.2

Attribute Value Designation for Case 1

Class Value	True Positive	True Negative	False Positive	False Negative
A	Yes	No	No	No
B	No	Yes	No	No
C	No	Yes	No	No

Table 4.3

Attribute Value Designation for Case 2

Class Value	True Positive	True Negative	False Positive	False Negative
A	No	No	No	Yes
B	No	No	Yes	No
C	No	Yes	No	No

- Accuracy: the proportion of correct classifier predictions over all classifier predictions.

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

- Error Rate: the proportion of incorrect classifier predictions over all classifier predictions.

$$ER = 1 - A = \frac{FP + FN}{TP + TN + FP + FN}$$

- Precision: the proportion of correctly predicted class outputs over all predicted class output cases.

$$P = \frac{TP}{TP + FP}$$

- Recall: the proportion of correctly predicted class outputs over all true class output cases.

$$R = \frac{TP}{TP + FN}$$

- F-Measure: the weighted harmonic mean of the precision and recall. When $n = 1$, the precision and recall are weighted evenly. By adjusting the value of n, we adjust our preference toward either the precision or recall metric. For example, $F_{0.5}$ weights the precision twice as much as the recall, whereas F_2 weights the recall twice as much as the precision. For our analysis, we use F_1.

$$F_n = \frac{(n^2 + 1)P \times R}{(n^2 \times P) + R}$$

In addition to the classification performance metrics, we also measure decision tree structural metrics. The structural metrics we use include:

- Node Count: the total number of nodes in a decision tree.

- Tree Depth: the number of branches taken to traverse a tree from its root to every leaf.

- Classification Complexity: the number of branches taken to traverse a tree from its root to the leaf used for classification, independent of classification correctness.

4.1.3 Competing Splitting Criteria

We compare the Rough Product splitting criterion to three popular splitting criteria used to construct decision trees: the Gini Index, Maximum Deviance Reduction (MDR), and Twoing Rule. Section 2.2.2 contains detailed information about each competing splitting criterion.

4.1.4   Tree Evaluation Graphical User Interface

In this section, we describe our EvalTreeGUI software tool, as shown in Figure 4.1. We use it to evaluate binary decision trees across various metrics. It uses MATLAB® to perform *k*-folds cross validation over a range of splitmins for any selected tree construction option. When the evaluation is complete, we use an advanced integrated plotting system to easily analyze the results.

4.1.4.1   File Menu

The File menu is the conduit for transferring data into and out of the EvalTreeGUI program. The Open command loads data from a properly formatted Excel spreadsheet, automatically determining categorical and numerical attributes. The program aborts loading if a categorical attribute is not detected. The Load and Save commands allow the user to restore and remember the current program state, respectively. These commands prove useful when evaluating enormous data sets. For convenience, the program state is stored to a MAT-file. The Print command sends an image of the EvalTreeGUI dialog to a specified printer, and the Close command exits the EvalTreeGUI program.

4.1.4.2   Status Panel

The Status panel communicates real time information about the EvalTreeGUI program status to the user, such as: success, error, and instructional messages; file load statistics (number of samples, number of features); and tree evaluation status (splitmin number, fold number, and percent complete).
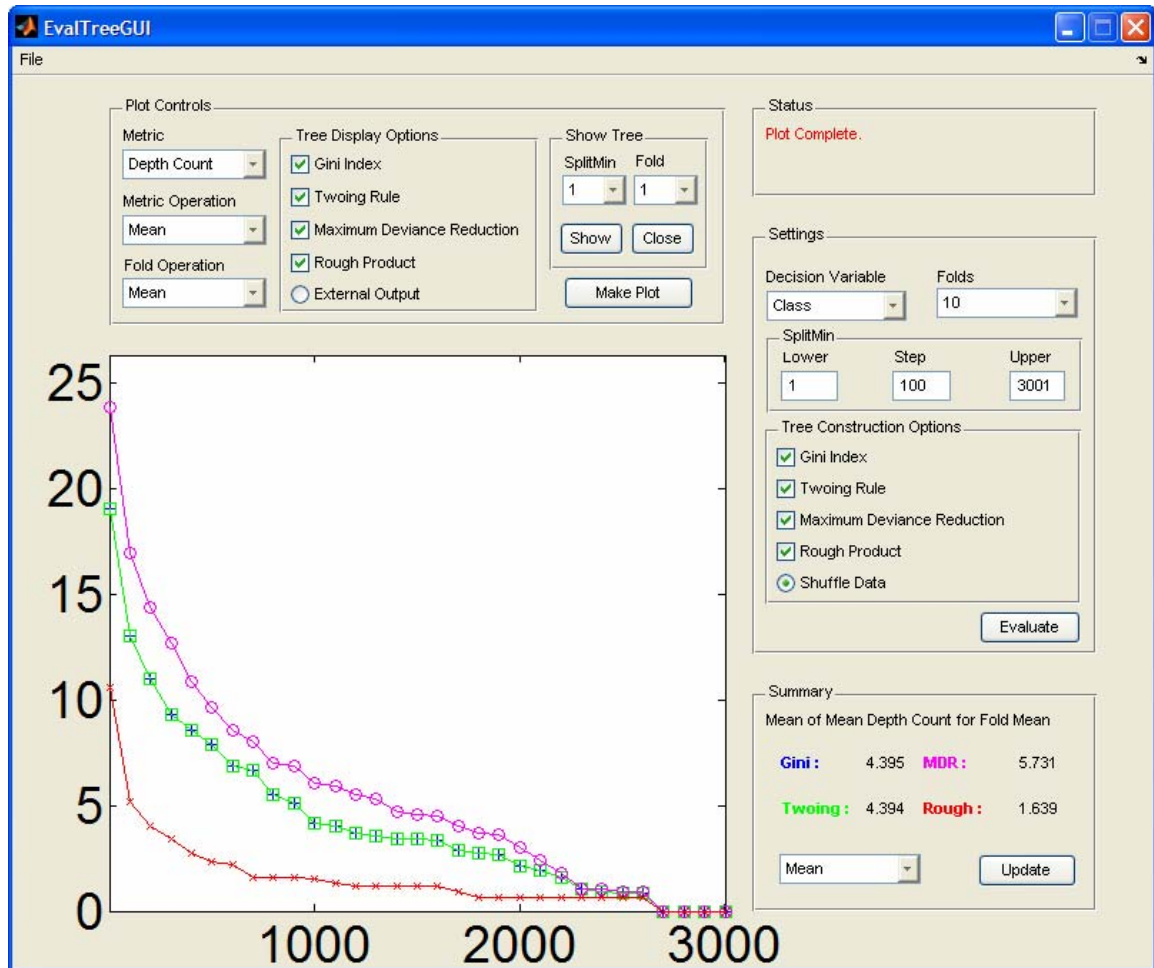
Figure 4.1: Screenshot of the EvalTreeGUI software. This application provides the mechanisms to load Excel data sets, evaluate decision tree classifiers across various metrics with *k*-folds cross validation, and analyze results with an integrated plotting system. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

4.1.4.3   Settings Panel

The Settings panel allows the user to set various parameters for the tree evaluation process. By clicking the Evaluate button, the program begins the cross validation process according to the set parameters, providing real time status updates in the Status panel.

The Decision Variable pop-up menu lists all categorical attributes in the loaded data. The user may choose any one of these categorical attributes for classification. By default, the program selects the first available categorical attribute.

The Folds pop-up menu lists the *k* values, ranging from two to ten, that can be set for the *k*-folds cross validation process. By default, the program selects "5."

The SplitMin sub-panel contains three textboxes that set the range and step interval of the splitmin parameter. The Lower and Upper textboxes define the splitmin lower and upper bound, respectively. The Step textbox defines the splitmin step interval. The program only accepts whole numbers greater than one in each field, and the value in Lower cannot be greater than that in Upper. The program automatically presets the textbox fields to the following values as the Open command executes:

$$Lower = 1$$

$$Upper = \left\lceil \frac{NumSamples \times (Folds - 1)}{Folds} \right\rceil$$

$$Step = \left\lceil Upper \times 0.1 \right\rceil$$

The Tree Construction Options sub-panel allows the user to select the splitting criteria to evaluate with the loaded data set. In addition, this sub-panel provides an

option to shuffle the data object order. An unselected shuffle option ensures that the

currently loaded data object order will not change during an evaluation.

4.1.4.4   Plot Controls Panel

The Plot Controls panel allows the user to analyze the results of the tree

evaluation in a variety of ways

The Show Tree sub-panel consists of a SplitMin pop-up menu, Fold pop-up

menu, Show button, and Close button. By clicking the "Show" button, the

EvalTreeGUI program calls the MATLAB® tree viewer (see Figure 4.2) to graphically

display the tree structure for a given splitting criterion at a set SplitMin and Fold. The

"Close" button closes all external MATLAB® windows.

By clicking the "Make Plot" button, the program generates a plot according to

the set parameters in the Metric pop-up menu, Metric Operation pop-up menu, Fold

Operation pop-up menu, and Tree Display Options panel.

The Metric pop-up menu allows the user to choose a classification or structural

metric for analysis. We define these metrics in Section 4.1.2.

The Metric Operation pop-up menu allows the user to apply a statistical

operation for a selected Metric. This choice combines the results of each decision value

with standard statistical functions, such as mean, median, minimum, and maximum.

For classification metrics, the user may also choose to analyze a particular decision

value individually with respect to the selected Metric. For example, for a classifier that

predicts benign and malignant tumors, the user may want to analyze the F-Measure for
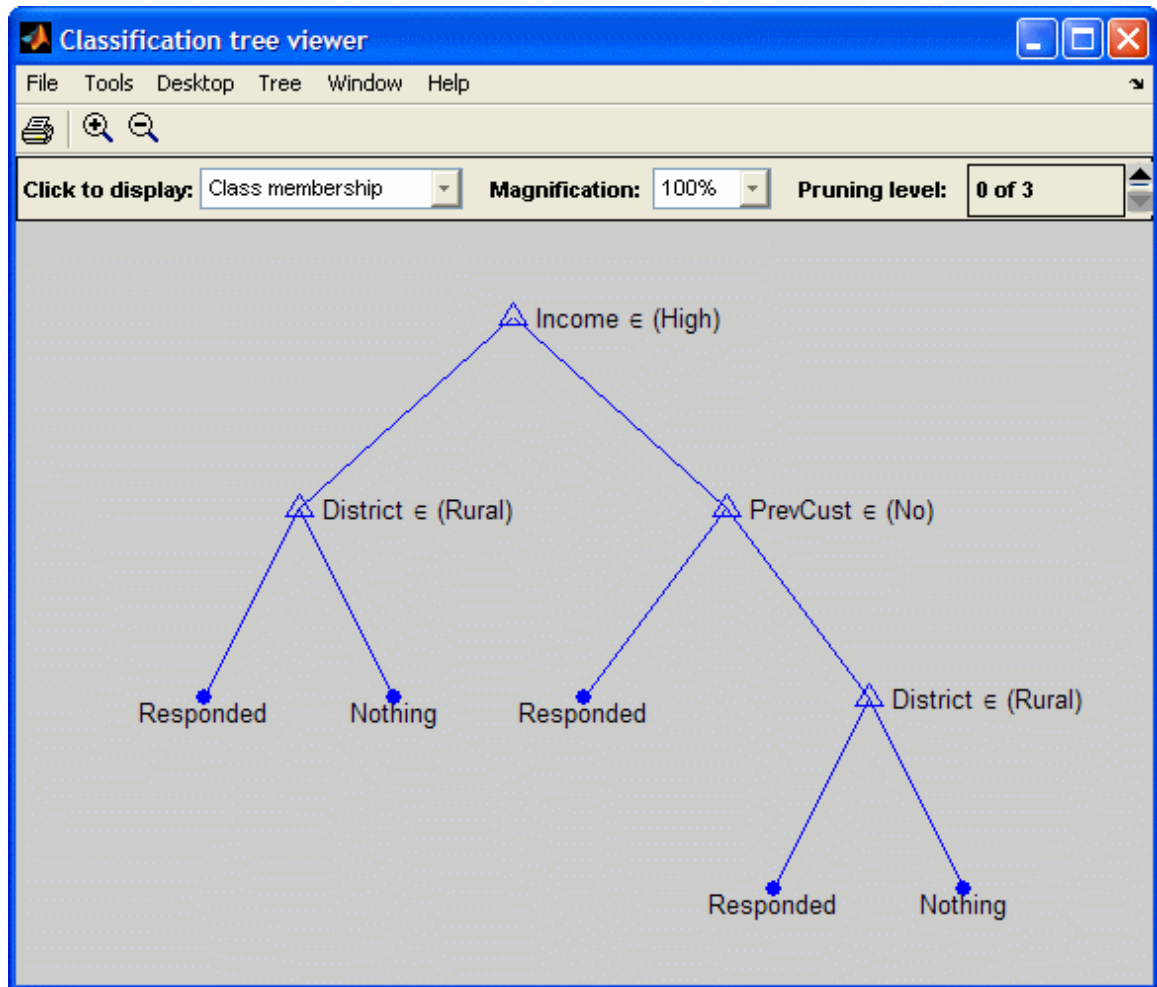
Figure 4.2: Screenshot of the Classification Tree Viewer. This program (treedisp.m) belongs to the MATLAB® Statistical Toolbox. It graphically displays a binary decision tree. Each internal node is labeled with its decision rule and each terminal node is labeled with the predicted value. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

only malignant predictions since incorrect malignant predictions would undoubtedly cause more tragedy for a patient.

The Fold Operation pop-up menu allows the user to apply a statistical operation to the selected Metric results from each fold. Like the Metric Operation, this choice combines the selected results with standard statistical functions, such as mean, median, minimum, and maximum. The user may also choose to analyze the selected results for an individual fold.

The Tree Display Options sub-panel allows the user to choose splitting criteria to analyze. These choices affect both the plot and tree viewer. In addition, the sub-panel provides an External Output radio button, which when selected, display the plot results in an independent figure window (see Figure 4.3). This feature is often useful when the user requires more tools to analyze the plot results or a printer-friendly version of plot for publication.

4.1.4.5   Summary Panel

The Summary Panel provides a global analysis of the points graphed in the plot. The user may choose to combine the plot points with standard statistical functions, such as mean, median, minimum, and maximum.

4.2   Experiments

We now present two sets of experiments to evaluate the Gini Index, Maximum Deviance Reduction (MDR), Twoing Rule, and Rough Product splitting criteria. We use the first set to characterize each criterion in response to various degrees of
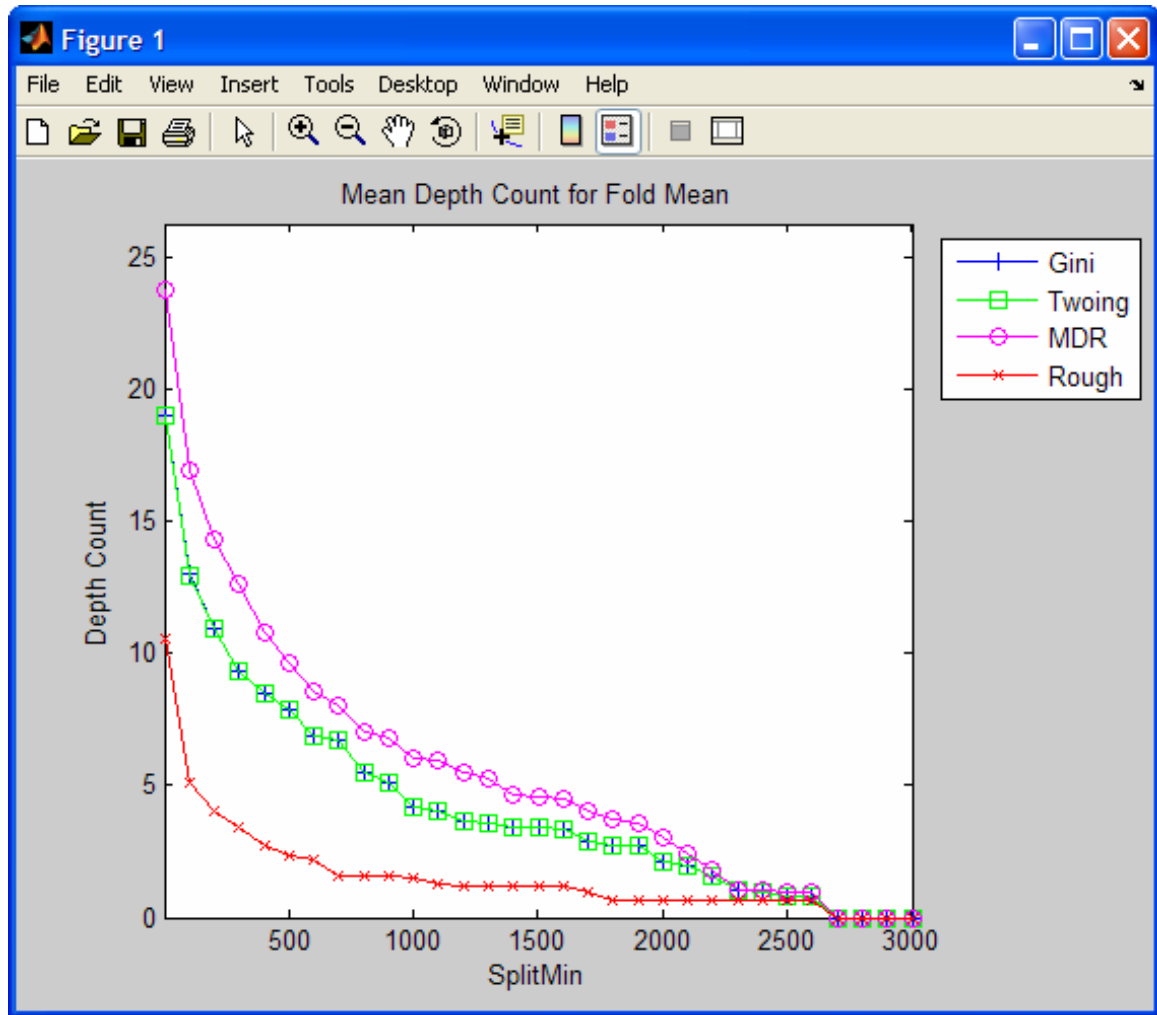
Figure 4.3: External Output Plot. To provide more flexibility and power during analysis, the user may configure the EvalTreeGUI software to automatically generate a plot in an independent MATLAB® figure window. Various tools within the figure window are available to assist in plot analysis. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

controlled noise. We use the second set to evaluate each splitting criterion on real-world data. The appendix contains tables that summarize the results for these experiments.

4.2.1   Gaussian Distribution Cluster Data Sets

In this set of experiments, we characterize the behavior of each splitting criterion in response to data containing various degrees of noise. These data sets are artificially generated three-dimensional points formed within Gaussian distributions:

$$(X,Y,Z) \approx \big(N(\mu,\sigma),N(\mu,\sigma),N(\mu,\sigma)\big).$$

We classify each point according to the distribution it came from. We give all classifiers the same training and testing sets without biases or cost functions. In addition, we perform all evaluations with a 10-folds cross validation across a range of splitmins.

4.2.1.1   Gaussian Experiment 1 – Perfectly Separable Case

In this experiment, we form two distinct noiseless classes that each contains 150 points, as shown in Figure 4.4(a). The points in the blue distribution come from $N(5,1)$, and those in the red come from $N(10,1)$. We aim to provide a baseline set of measurements and ensure that the splitting criteria can easily separate the classes.

As expected, each splitting criterion produced decision trees with the same structure, as shown in Figure 4.4(b, c, d, e, f, g, h, i). However, the choice of the splitting attribute value produced minor variances in the classification metrics. For real-valued attributes A, the Gini Index, Twoing Rule, and MDR implementations

51
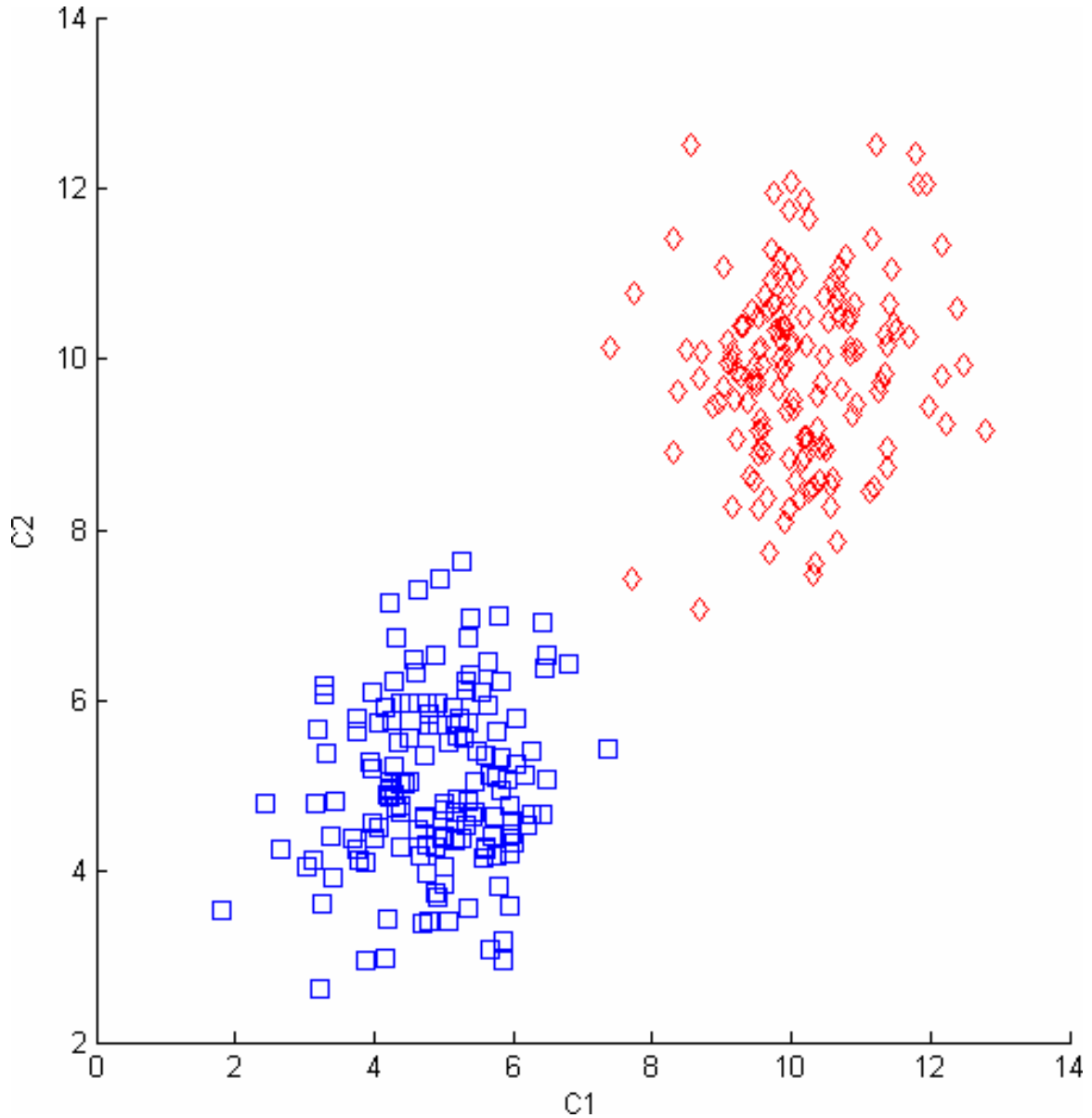
(a)

Figure 4.4: Gaussian Experiment 1. The data set (a) is constructed such that 150 blue three-dimensional points come from $N(5,1)$ and 150 red three-dimensional points come from $N(10,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.4 – Continued
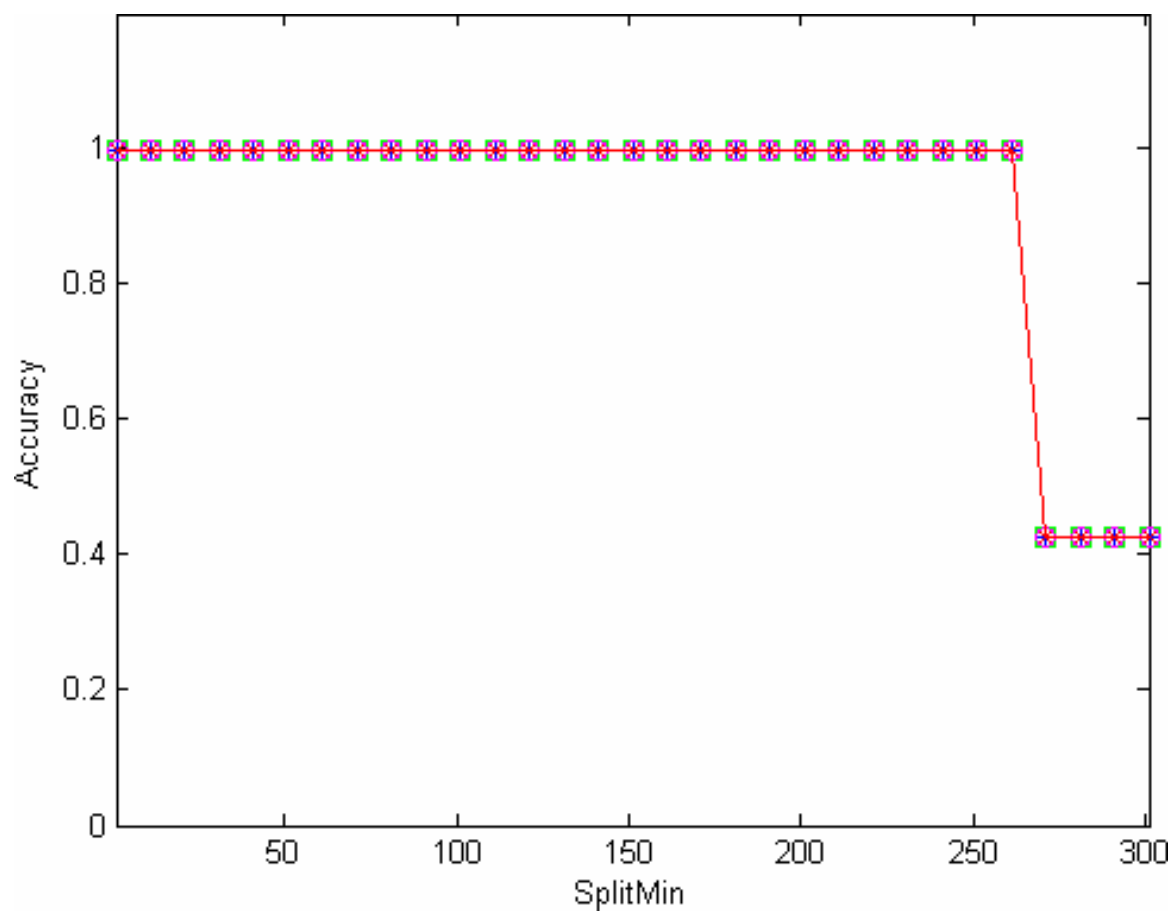
(b)

Figure 4.4 – Continued
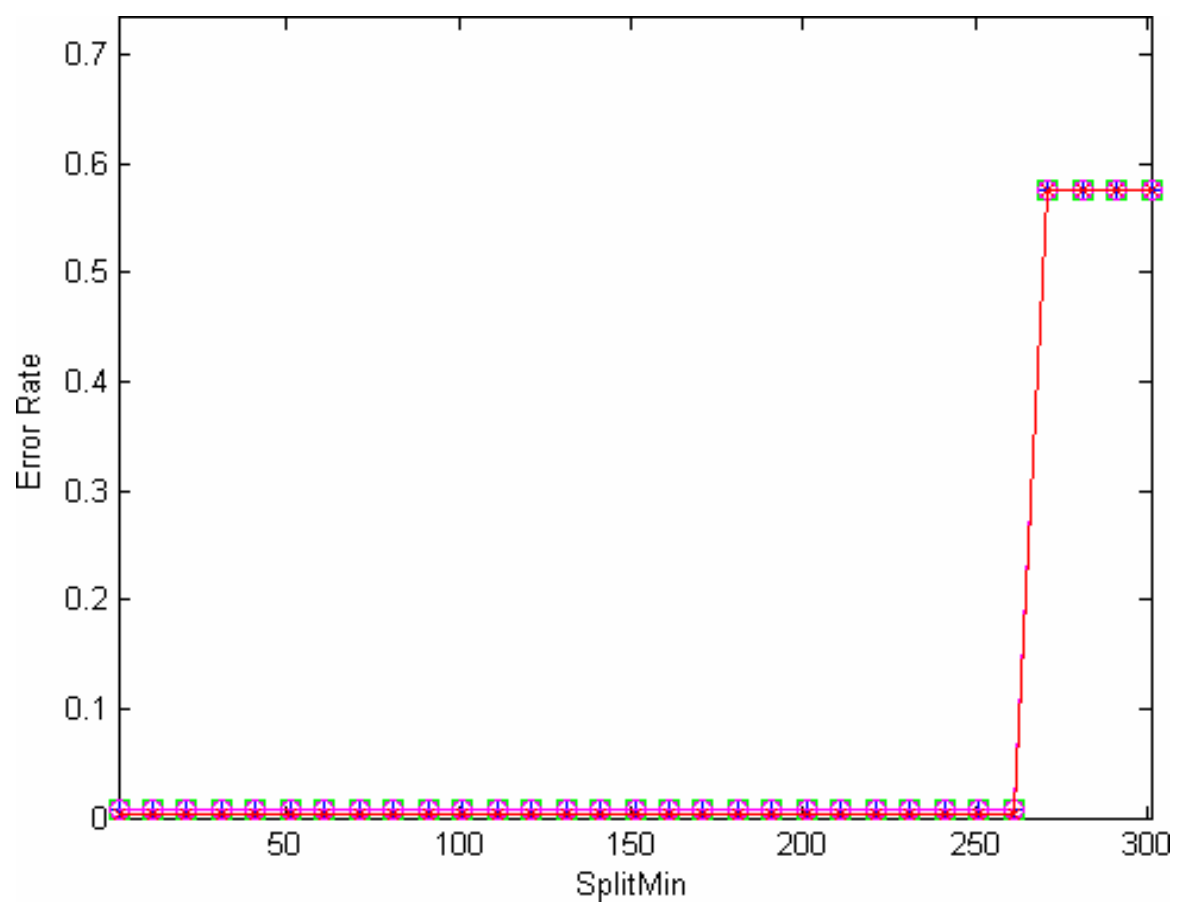
(c)

Figure 4.4 – Continued
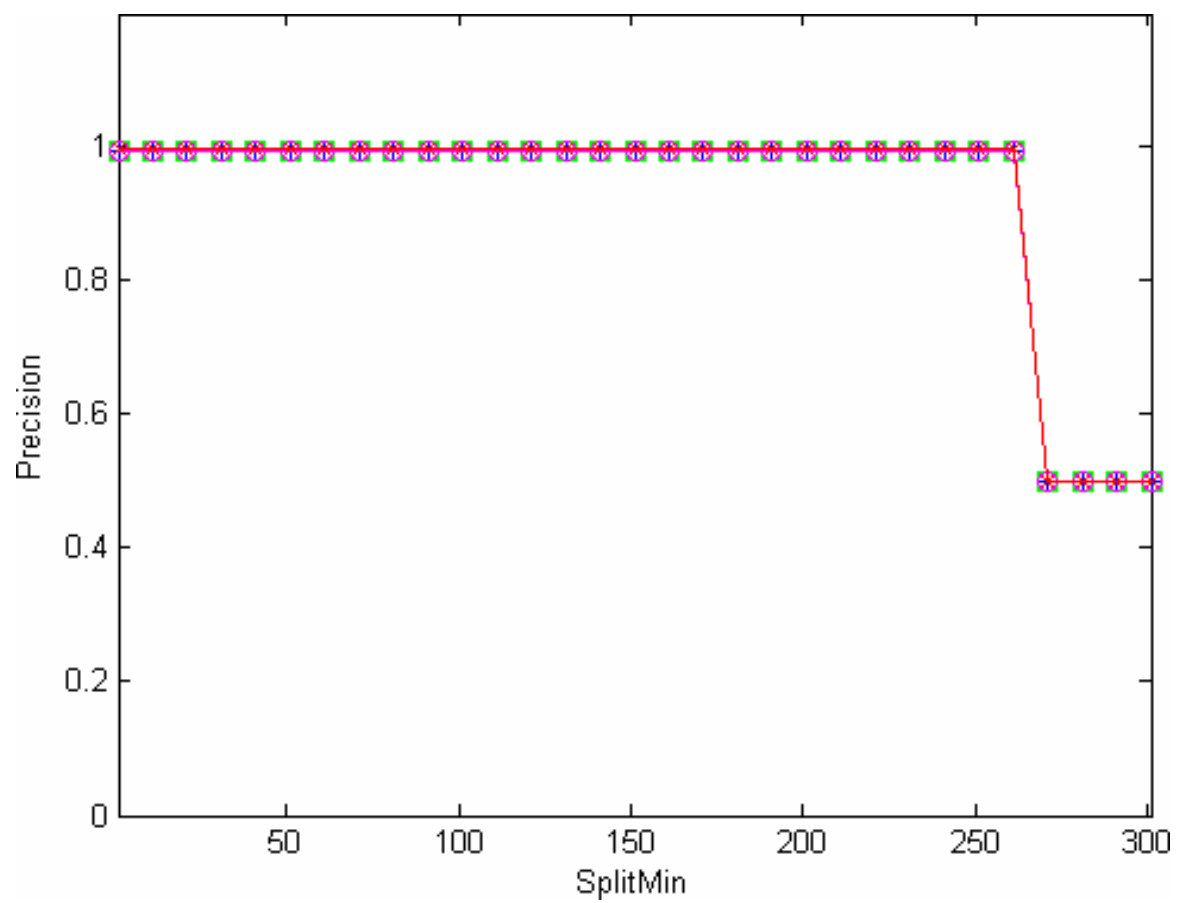
(d)

Figure 4.4 – Continued

(e)

56
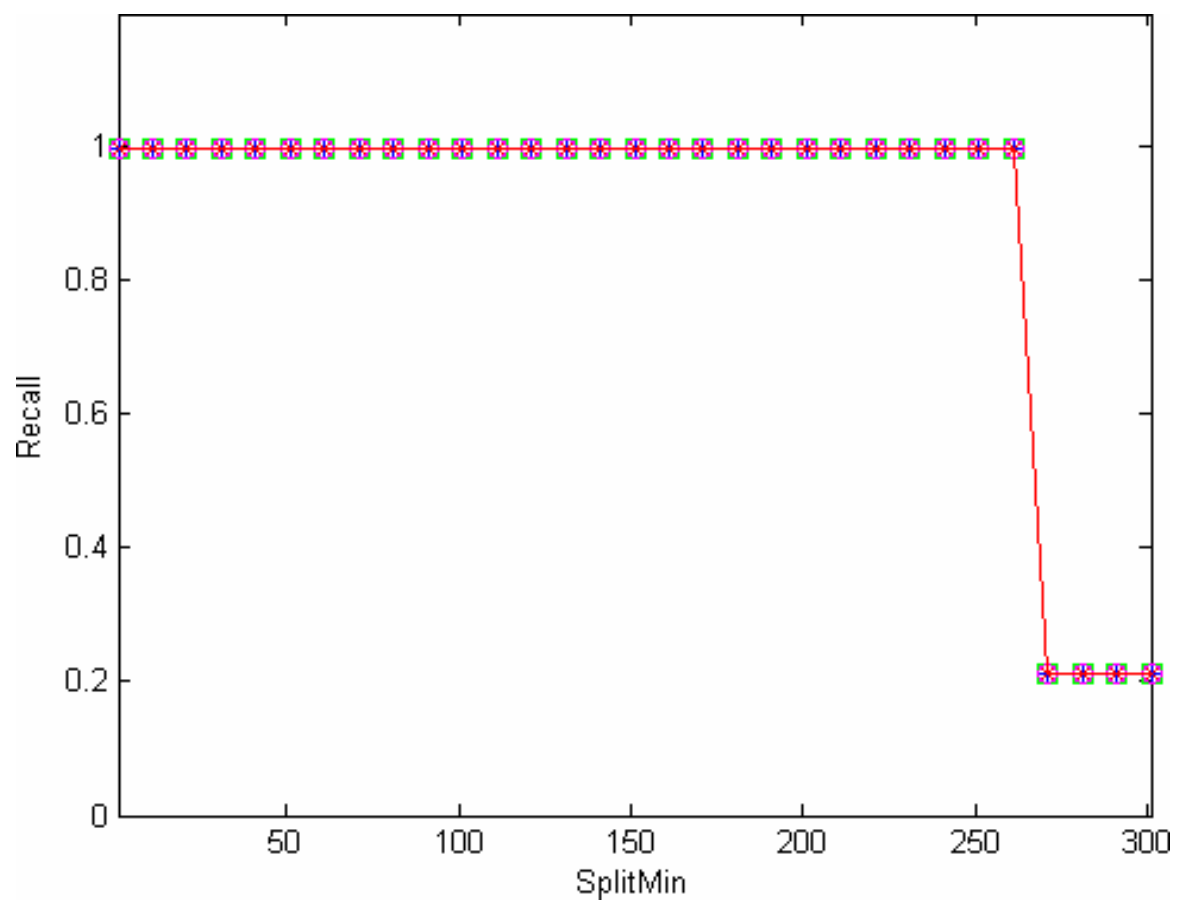
Figure 4.4 – Continued

(f)

57
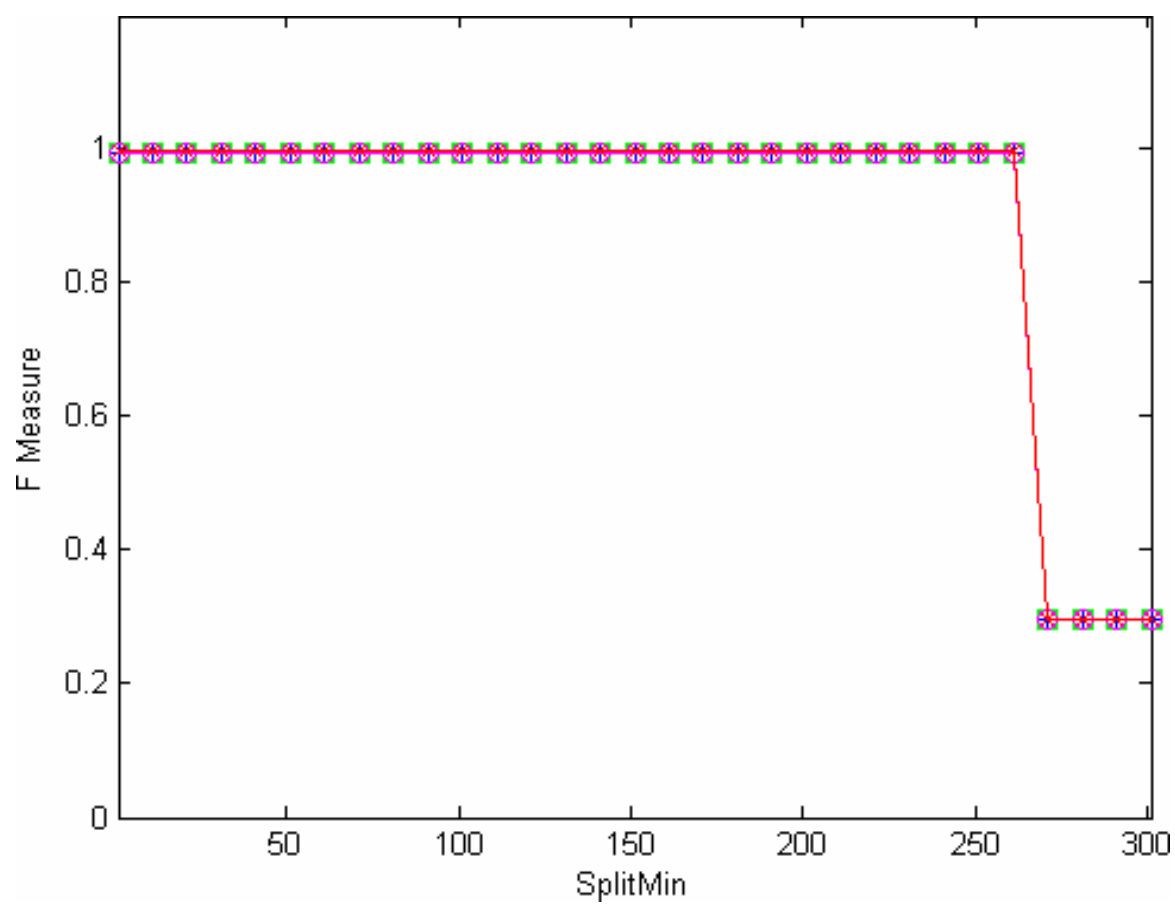
Figure 4.4 – Continued
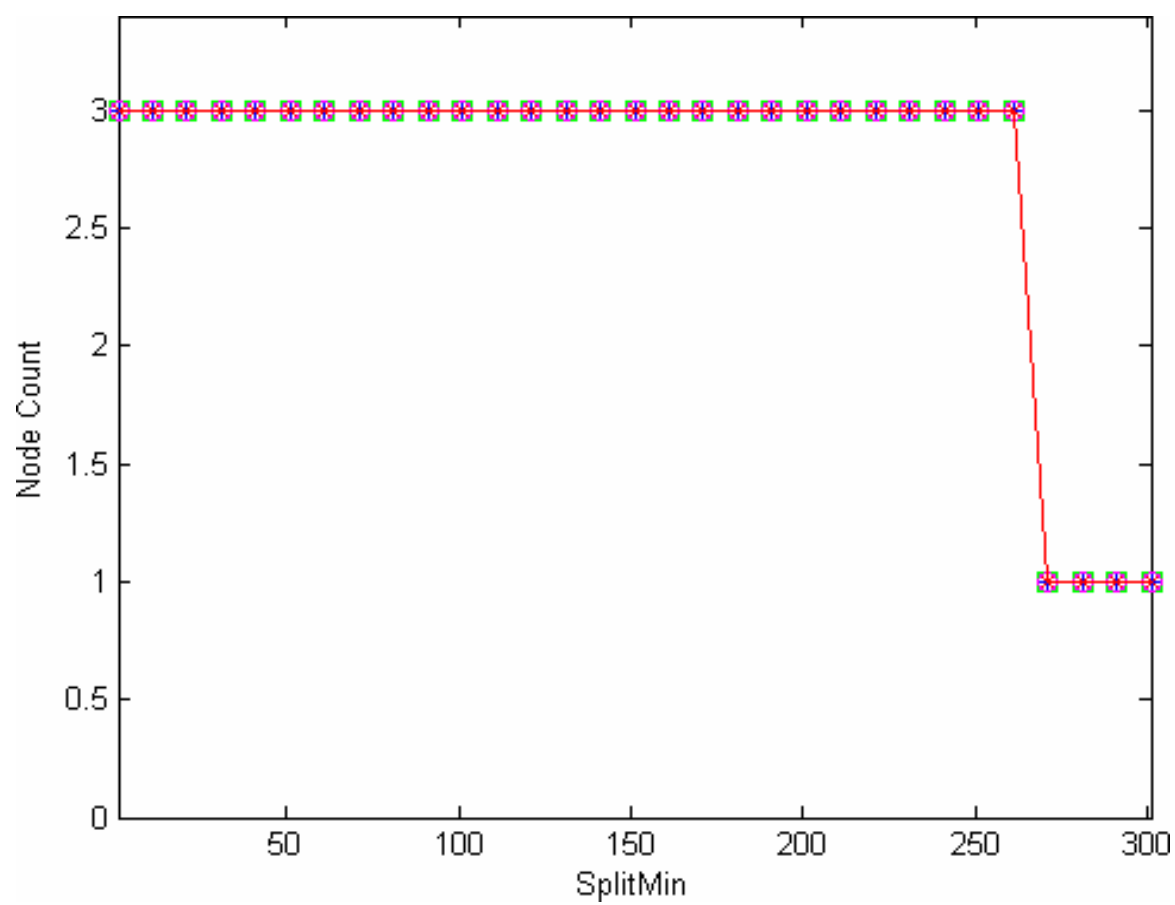
(g)

Figure 4.4 – Continued
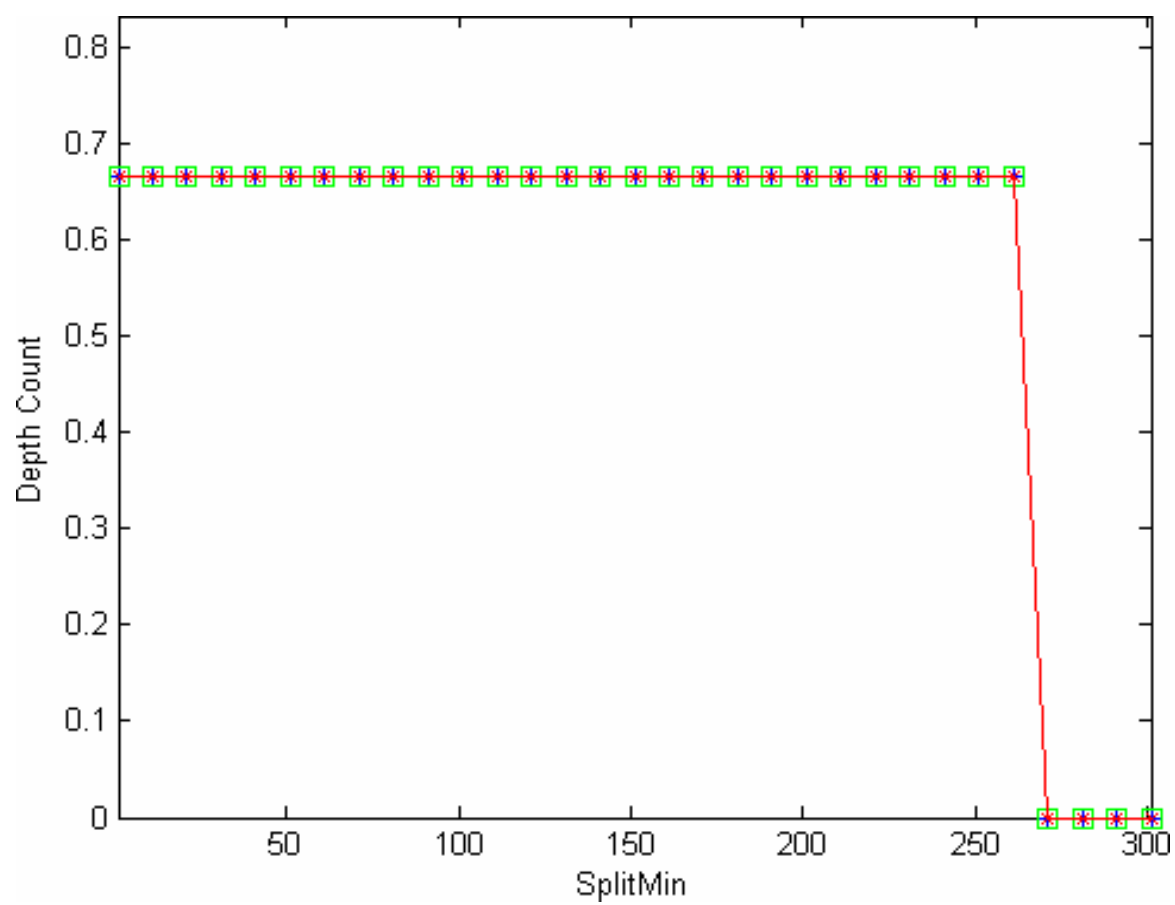
(h)

Figure 4.4 – Continued
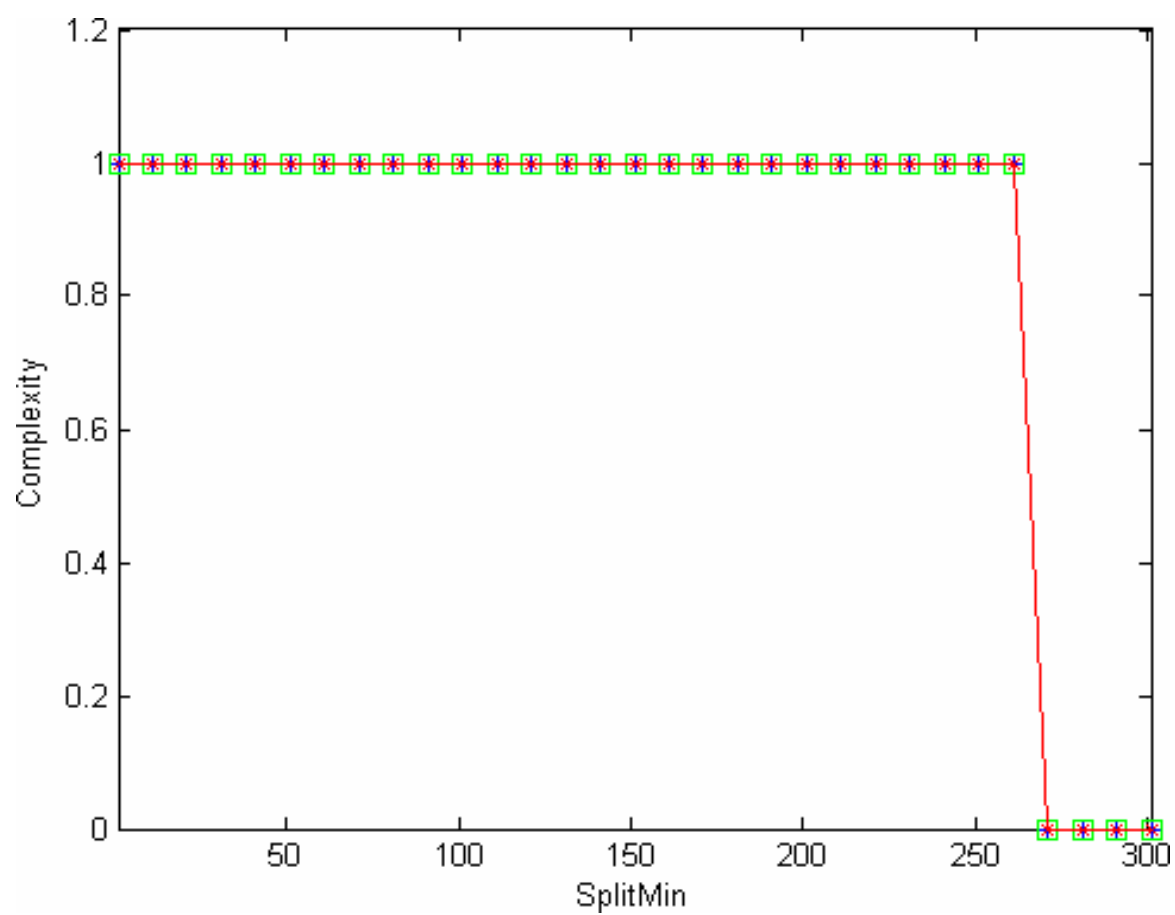
(i)

calculate the split value *s* as the midpoint of $A_l < A_s < A_u$ that leads to the same

maximum impurity reduction. The Rough Product implementation, on the other hand,

arbitrarily chooses either $A_s = A_l$ or $A_s = A_u$. Theoretically, we have little reason to

favor one split selection method over another, but the Rough Product implementation is

slightly simpler.

4.2.1.2 Gaussian Experiment 2a – Slightly Noisy Case with Low Density

In this experiment, we position the Gaussian distribution clusters closer to each

other, such that they slightly overlap. The blue distribution contains 150 points that

belong to $N(5,1)$, whereas the red distribution contains 150 points that belong to $N(8,1)$,

as shown in Figure 4.5(a). By introducing noise between the classes, we naturally

expect the classification metric results to decrease relative to the baseline, and the

structural metric results to increase relative to the baseline. We aim to determine how

drastic these changes are.

The results for the classification metrics, shown in Figure 4.5(b, c, d, e, f),

illustrate that, despite some variance, all splitting criteria perform similarly, and with a

very high classification accuracy. Compared to the baseline experiment, we see only a

small reduction in classification accuracy, confirming that the splitting criteria can

easily separate the classes despite the noise. Structurally, the Gini Index, Twoing Rule,

and Rough Product splitting criteria produce nearly equivalent decision trees, with

differences caused only by their split value selection methods. However, the MDR

splitting criterion generates completely different trees with a noticeably lower mean

node count, depth, and complexity, as shown in Figure 4.5(g, h, i).
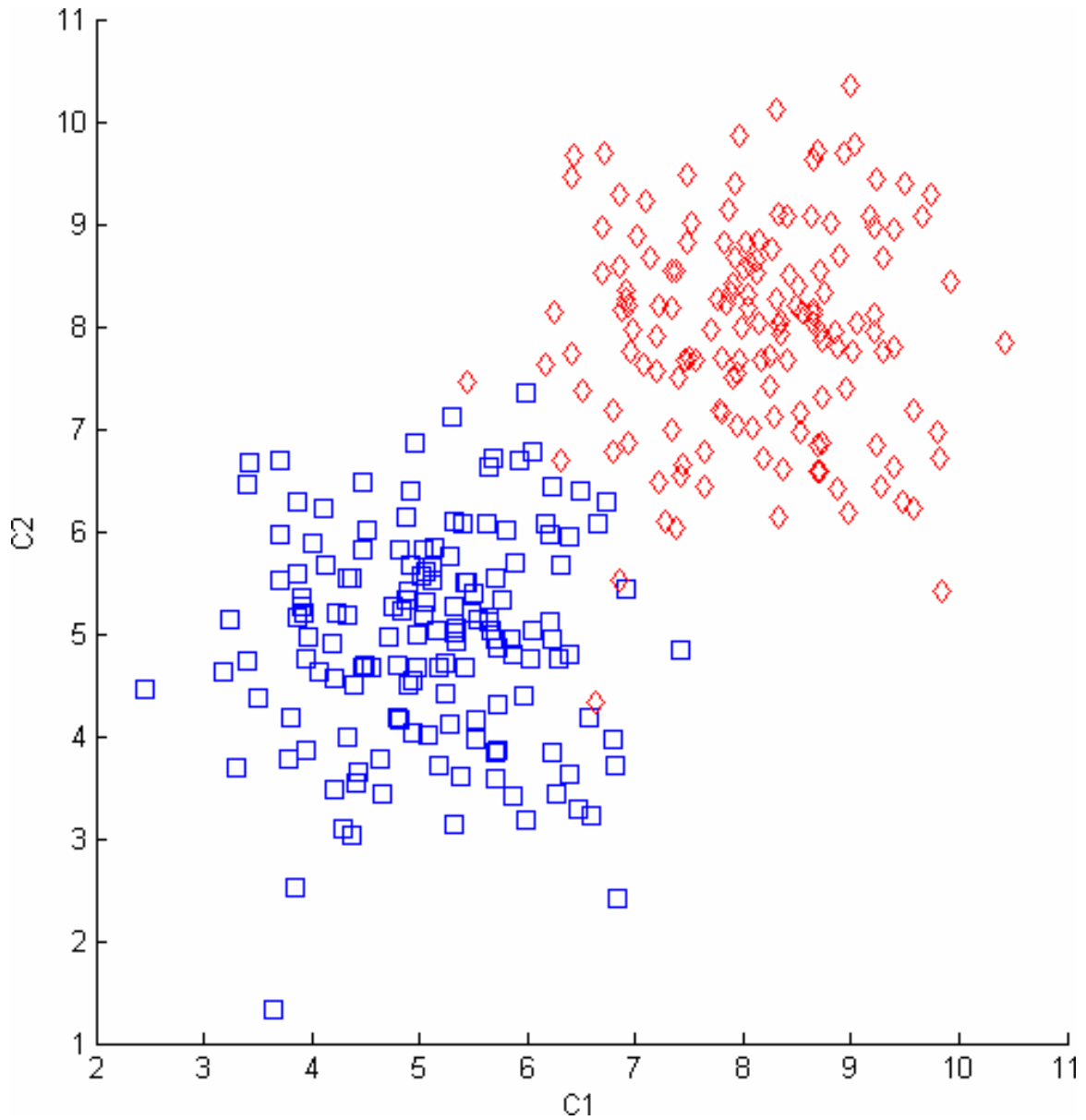
(a)

Figure 4.5: Gaussian Experiment 2a. The data set (a) is constructed such that 150 blue three-dimensional points come from $N(5,1)$ and 150 red three-dimensional points come from $N(8,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.5 – Continued
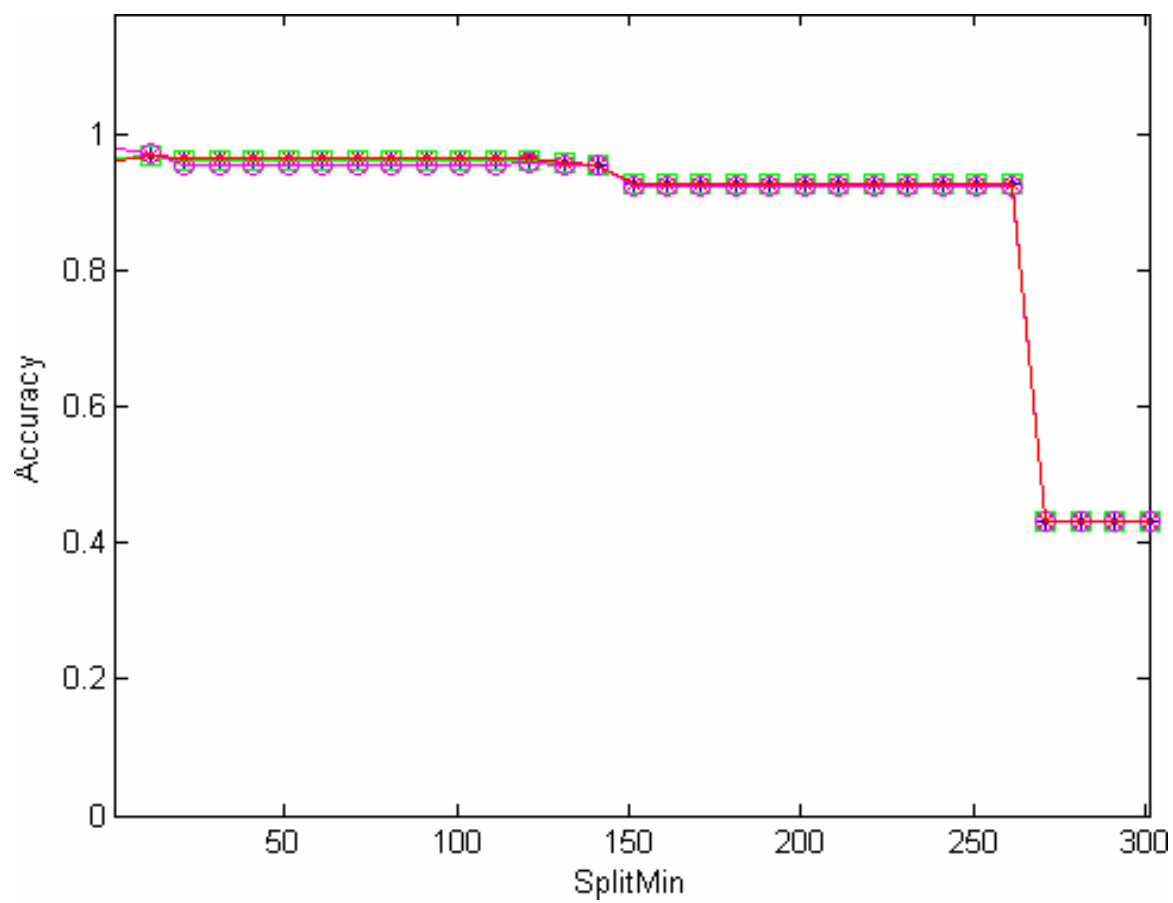

(b)

Figure 4.5 – Continued
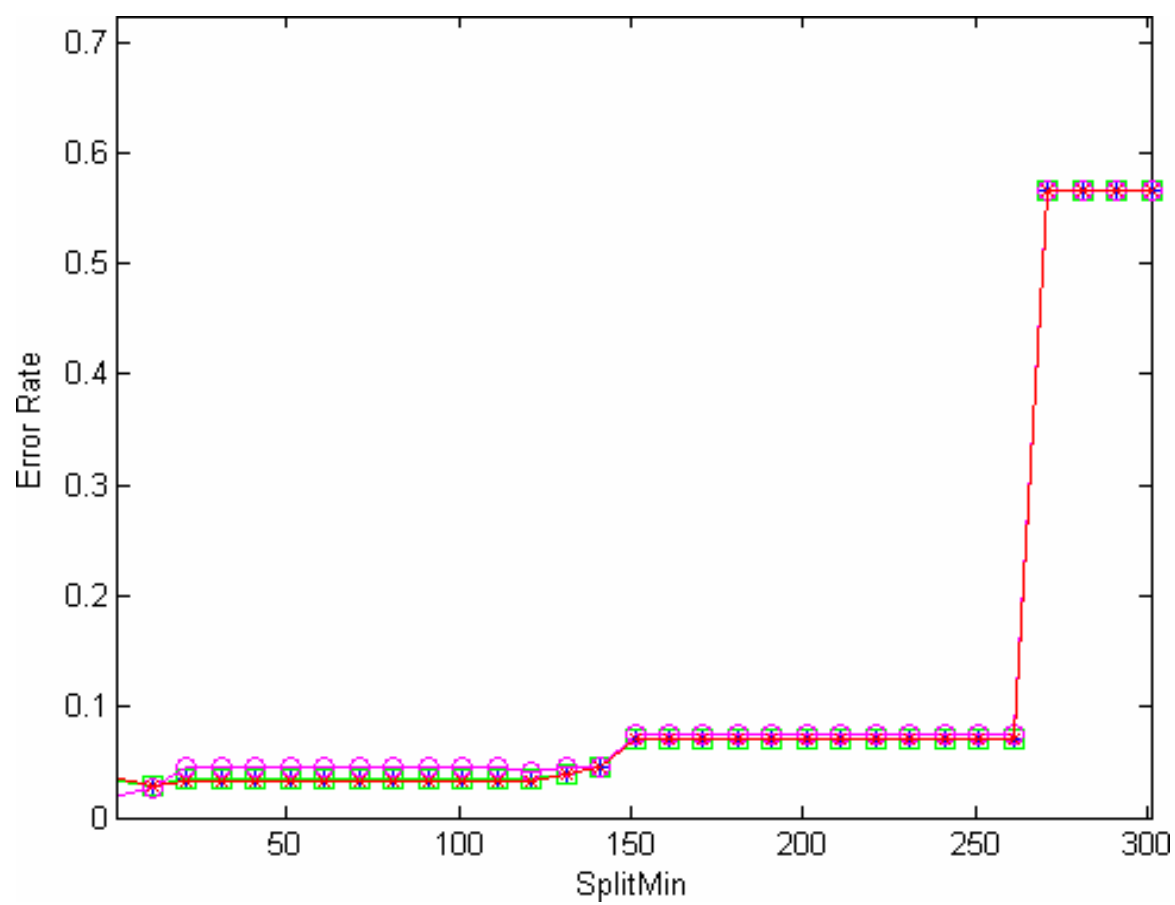

(c)

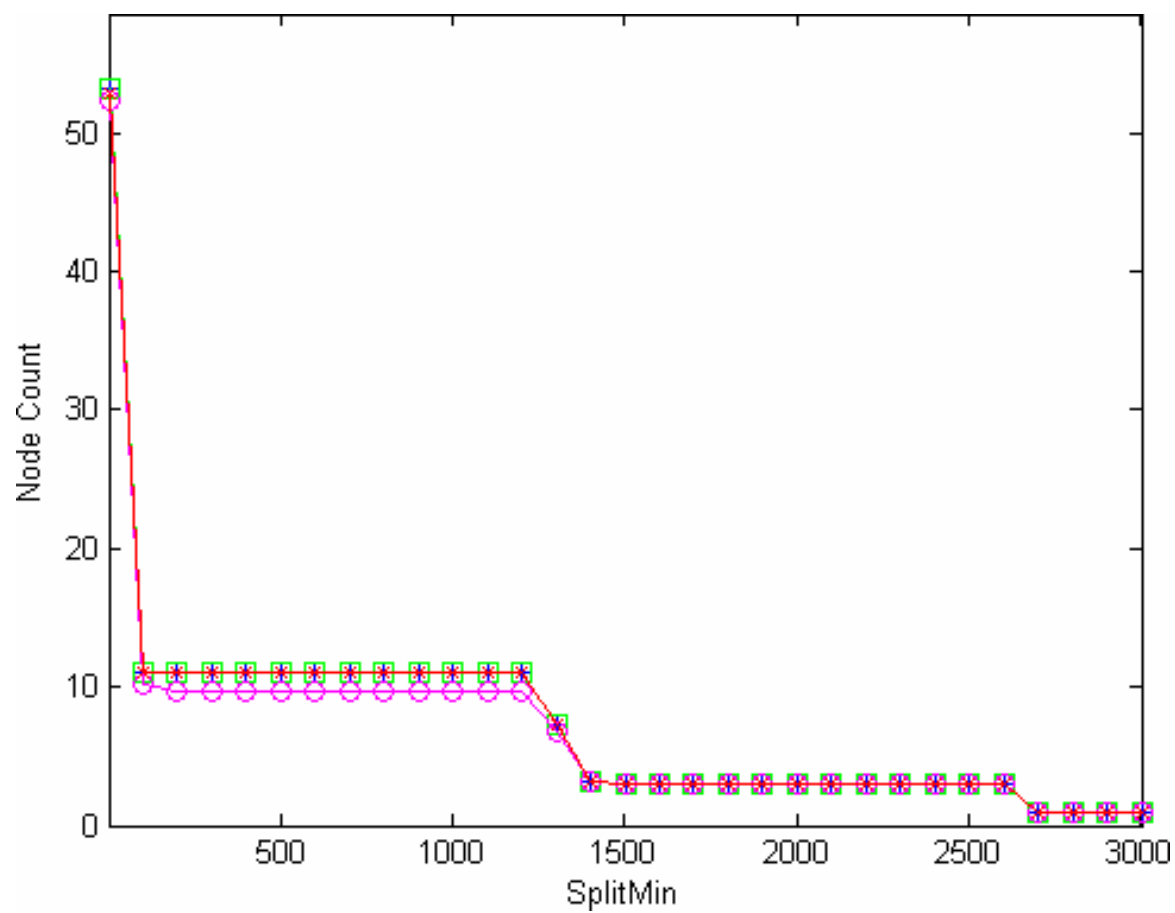Figure 4.5 – Continued
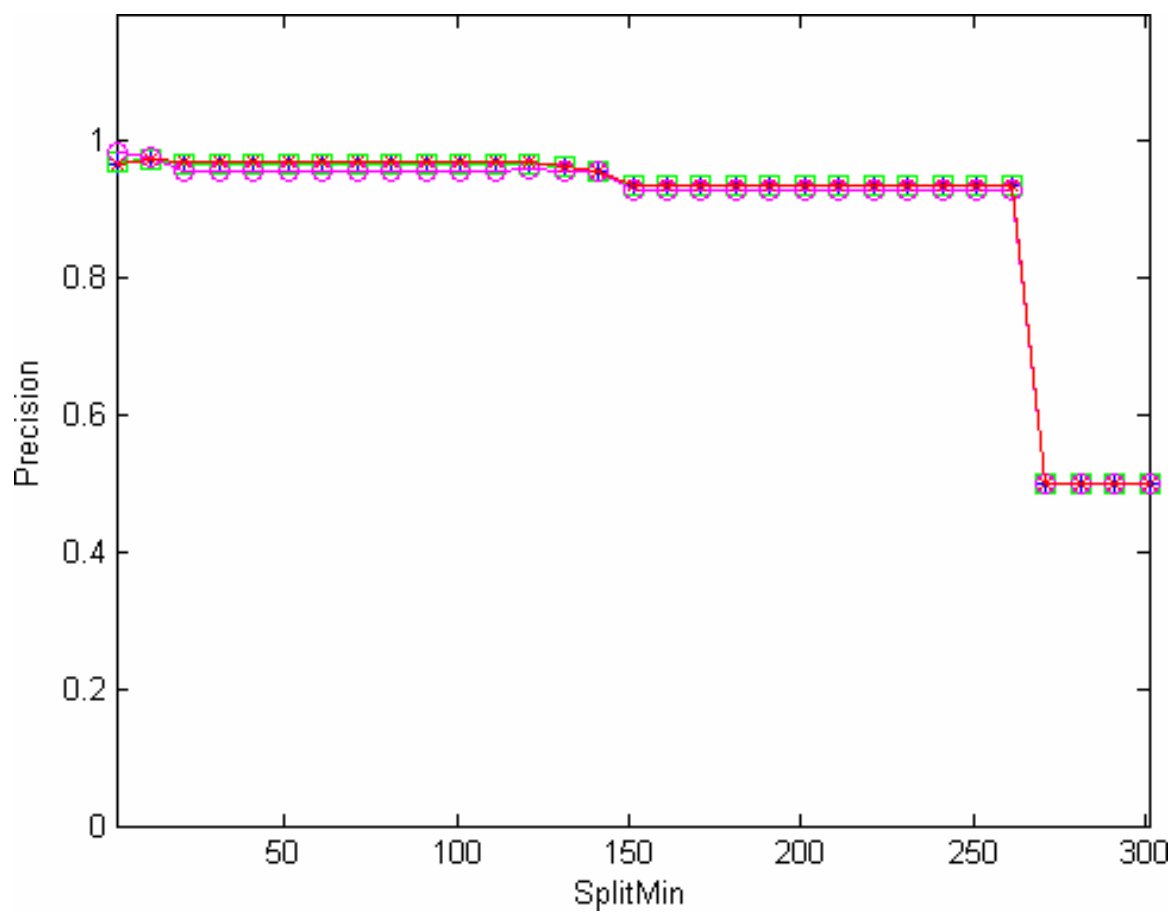
(d)

Figure 4.5 – Continued
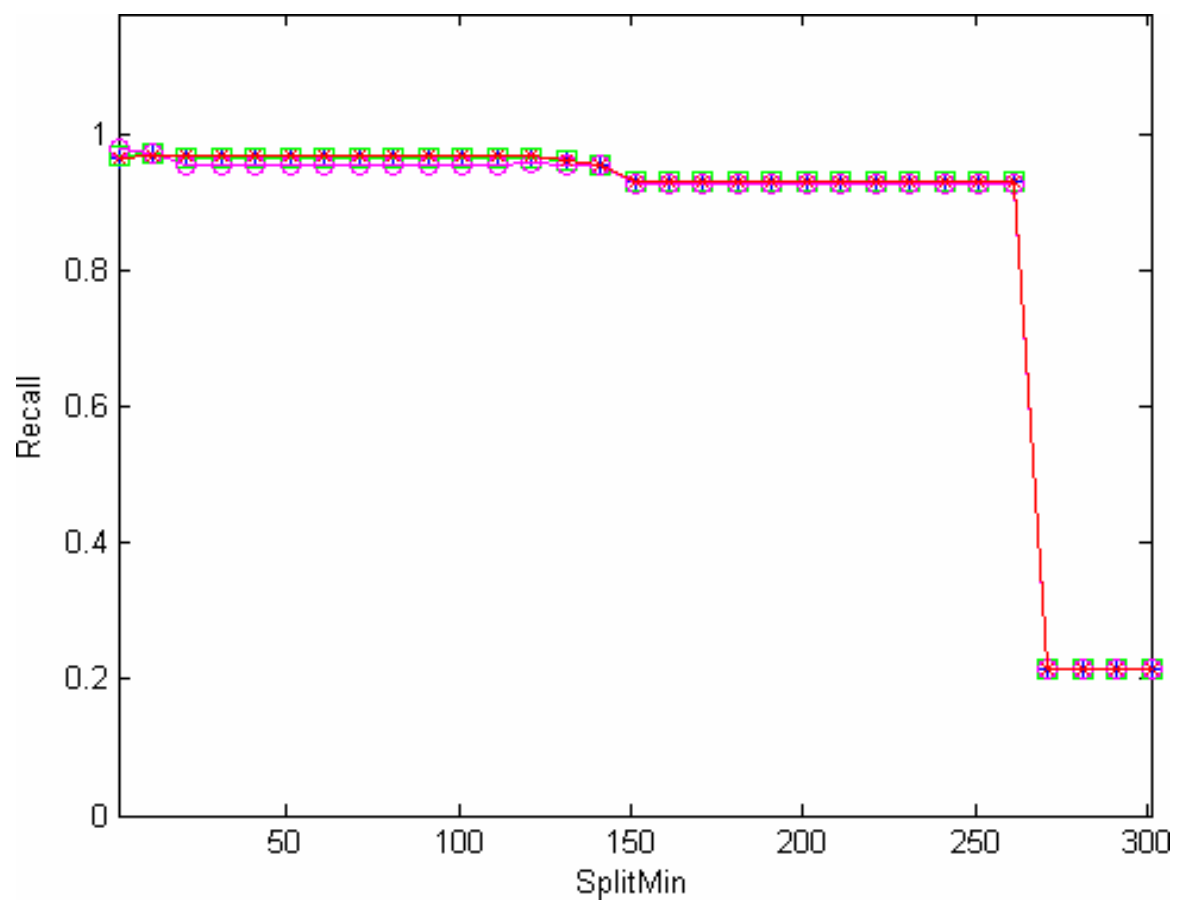

(e)

Figure 4.5 – Continued
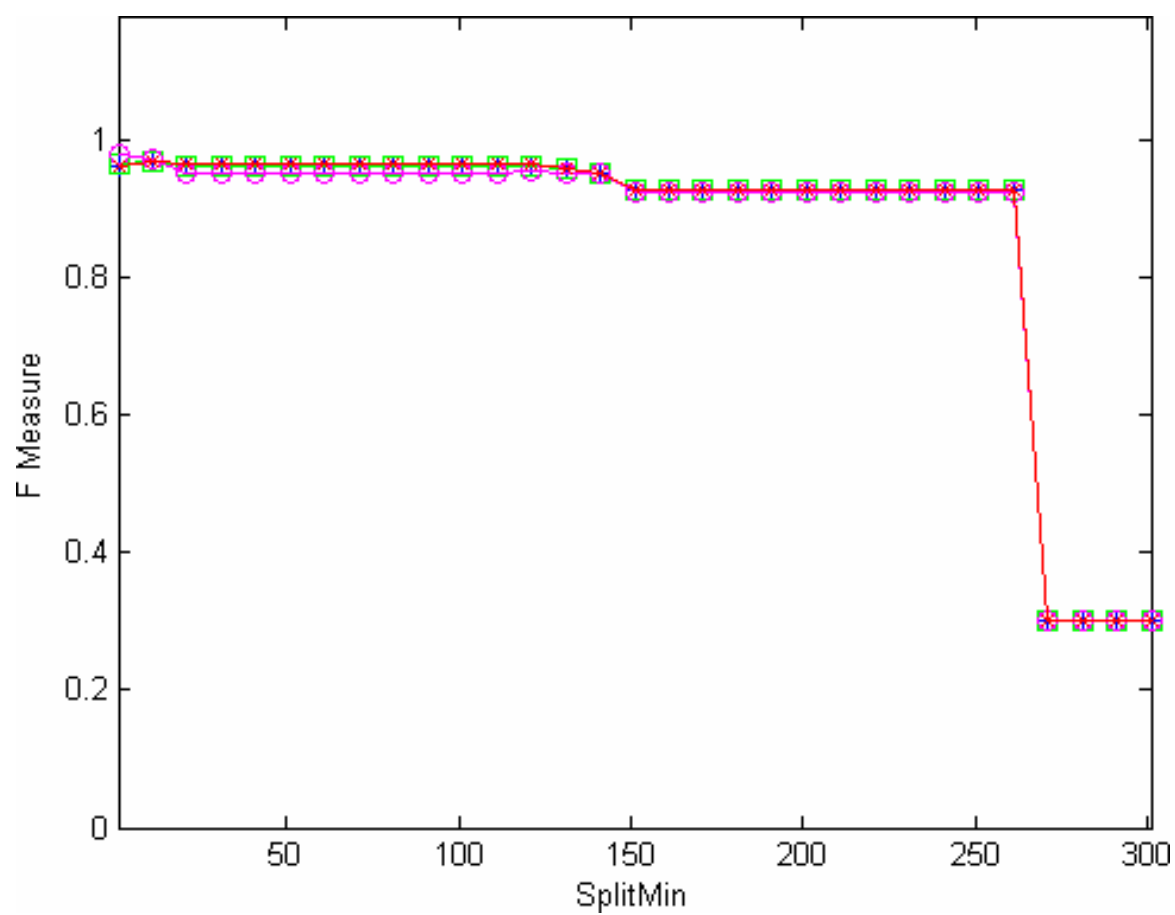

(f)

Figure 4.5 – Continued
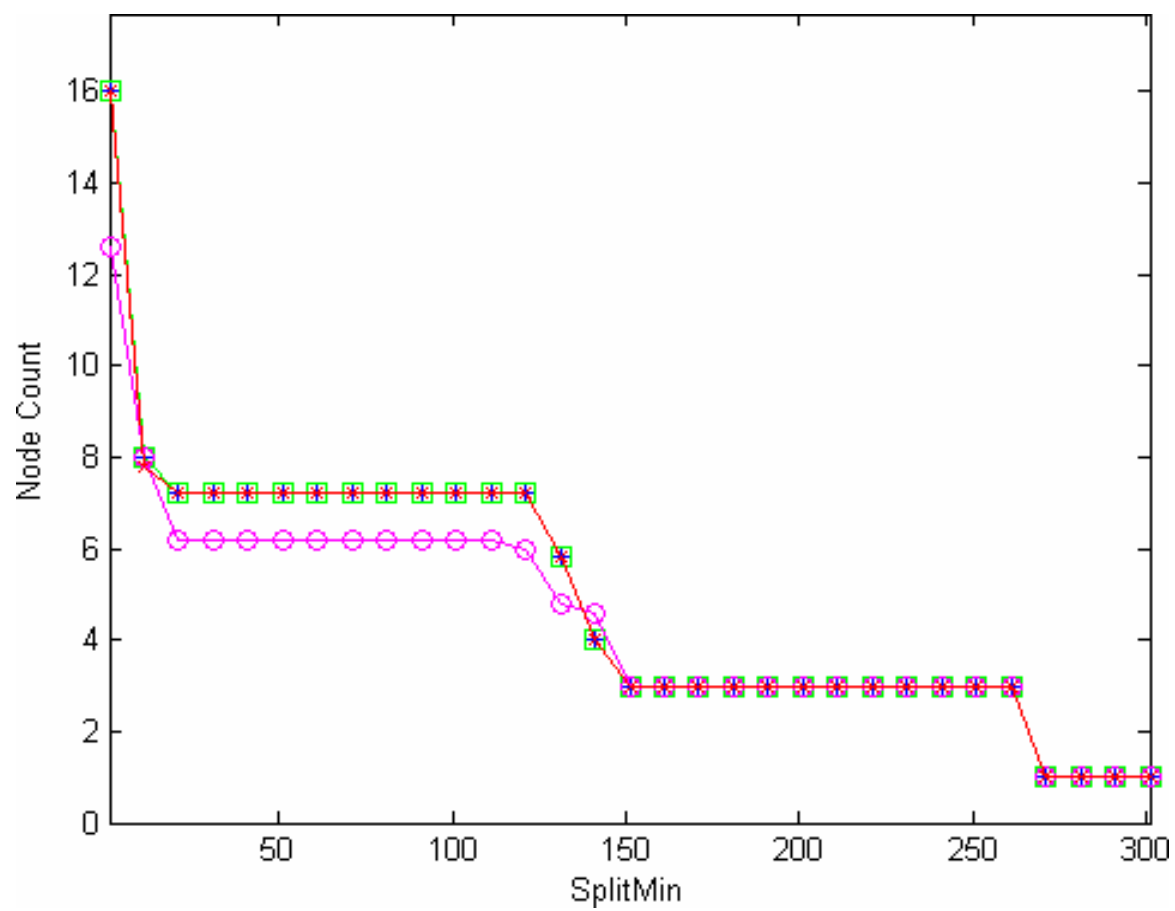
(g)

Figure 4.5 – Continued
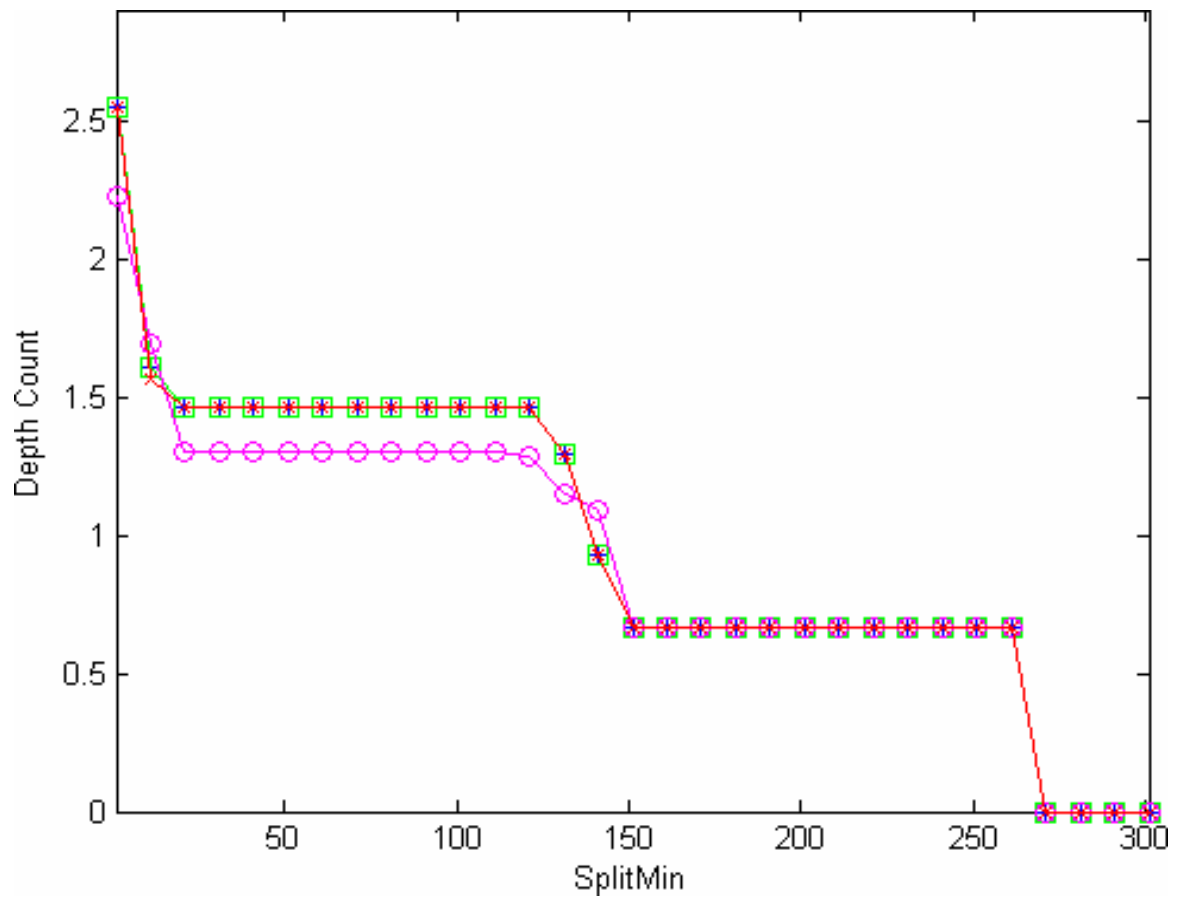
(h)

Figure 4.5 – Continued
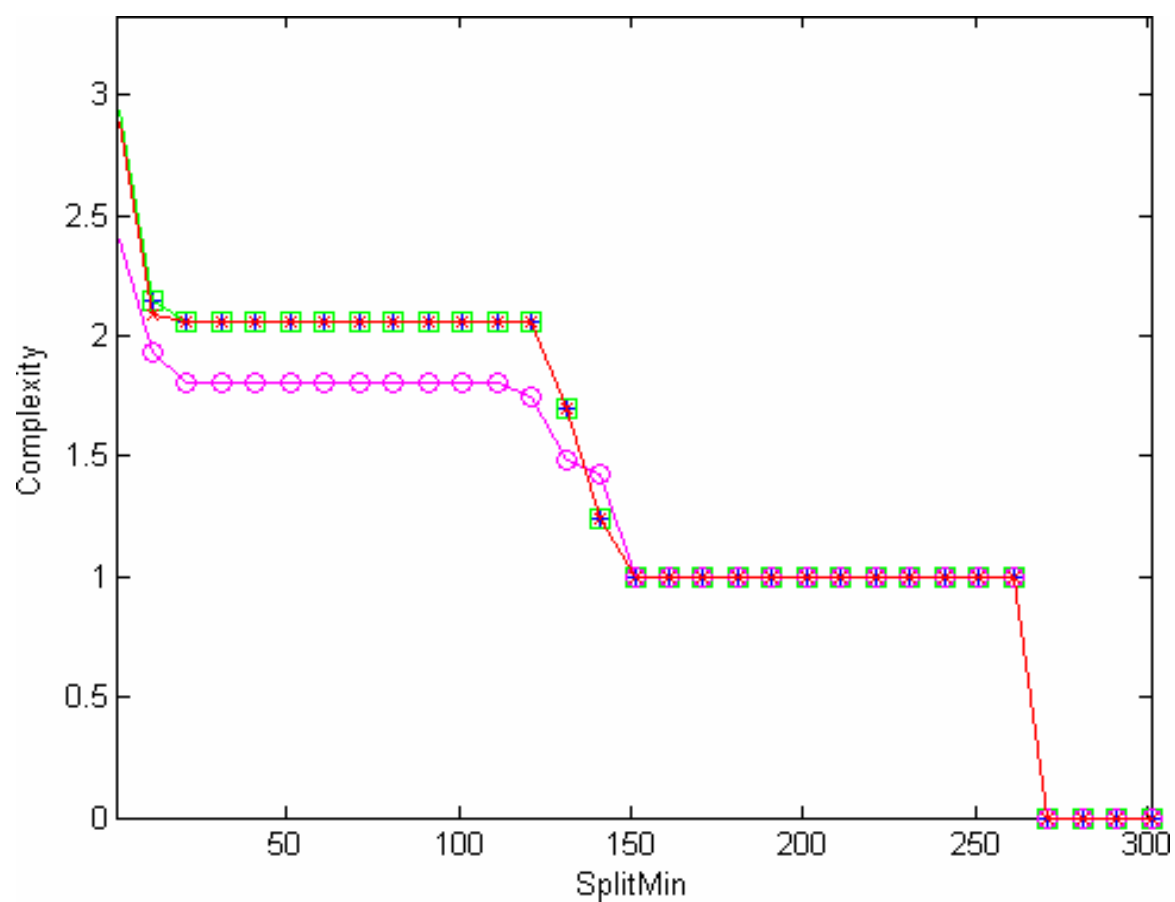
(i)

4.2.1.3 Gaussian Experiment 2b – Slightly Noisy Case with High Density

As in Gaussian Experiment 2a, the Gaussian distributions slightly overlap. However, we increase the density of each distribution by an order of magnitude, such that each decision class contains 1500 data points. The blue distribution contains points that belong to $N(5,1)$, and the red distribution contains points that belong to $N(8,1)$, as shown in Figure 4.6(a). We aim to determine whether a noticeable difference exists between the results in this experiment and Gaussian Experiment 2a. We expect the resultant decision trees to be very similar to those produced in Gaussian Experiment 2a because the samples in both experiments are representative of each other. In addition, we measure each splitmin at the same interval size relative to the total number of points as in Gaussian Experiment 2a.

The classification metric results, shown in Figure 4.6(b, c, d, e, f), are highly correlated to those in Gaussian Experiment 2a. Structurally, however, the decision trees for each splitting criterion become noticeably larger. Despite the increase in size, the resultant plots, shown in Figure 4.6(g, h, i), retain shapes that are similar to those in Experiment 2a. The Gini Index, Twoing Rule, and Rough Product splitting criteria produce nearly equivalent decision trees, with differences caused only by their split value selection methods, as in Experiment 2a. Moreover, as before, the MDR splitting criterion generates completely different trees with a noticeably lower mean node count, depth, and complexity.
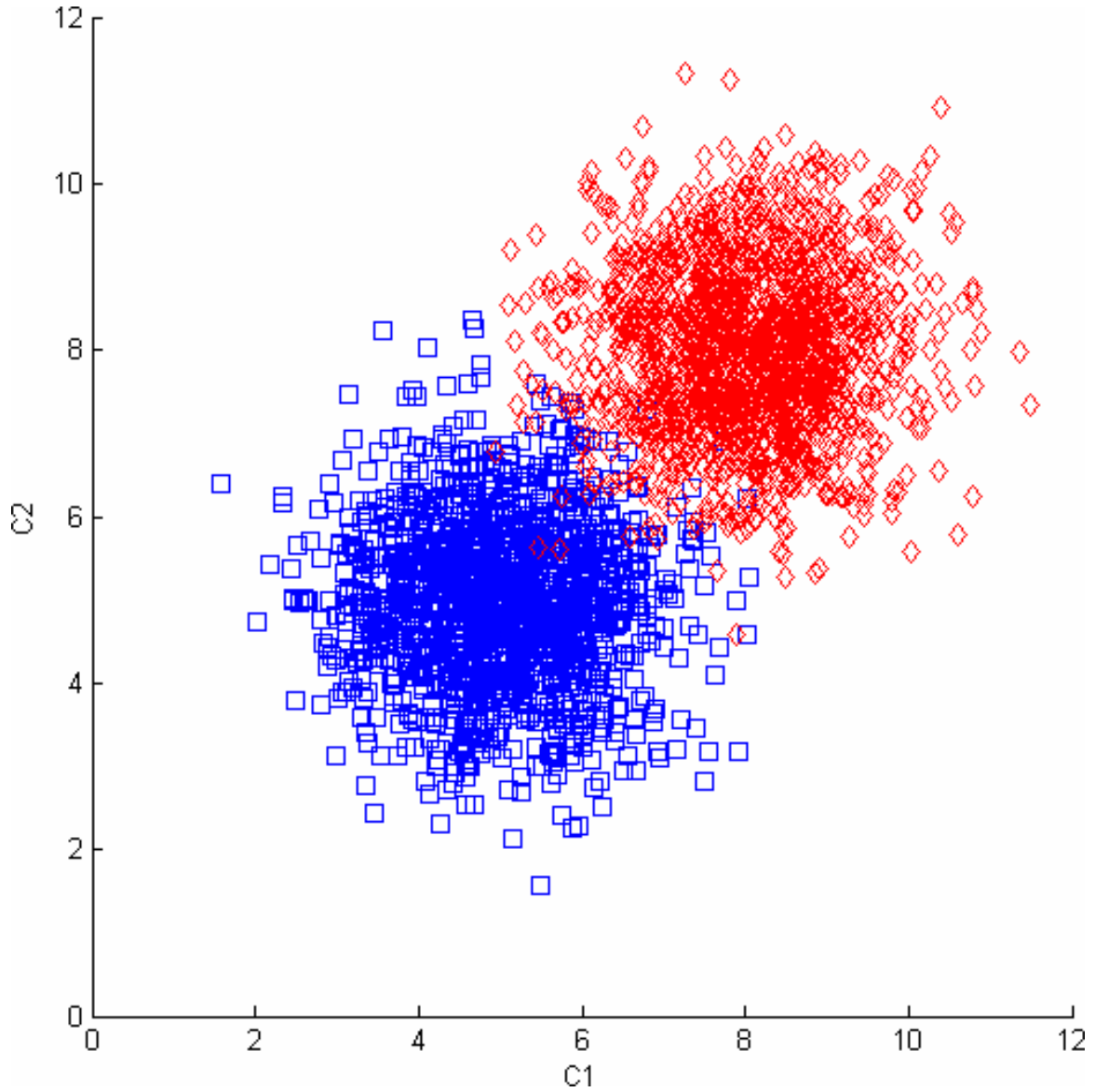
(a)

Figure 4.6: Gaussian Experiment 2b. The data set (a) is constructed such that 1500 blue three-dimensional points come from $N(5,1)$ and 1500 red three-dimensional points come from $N(8,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.6 – Continued
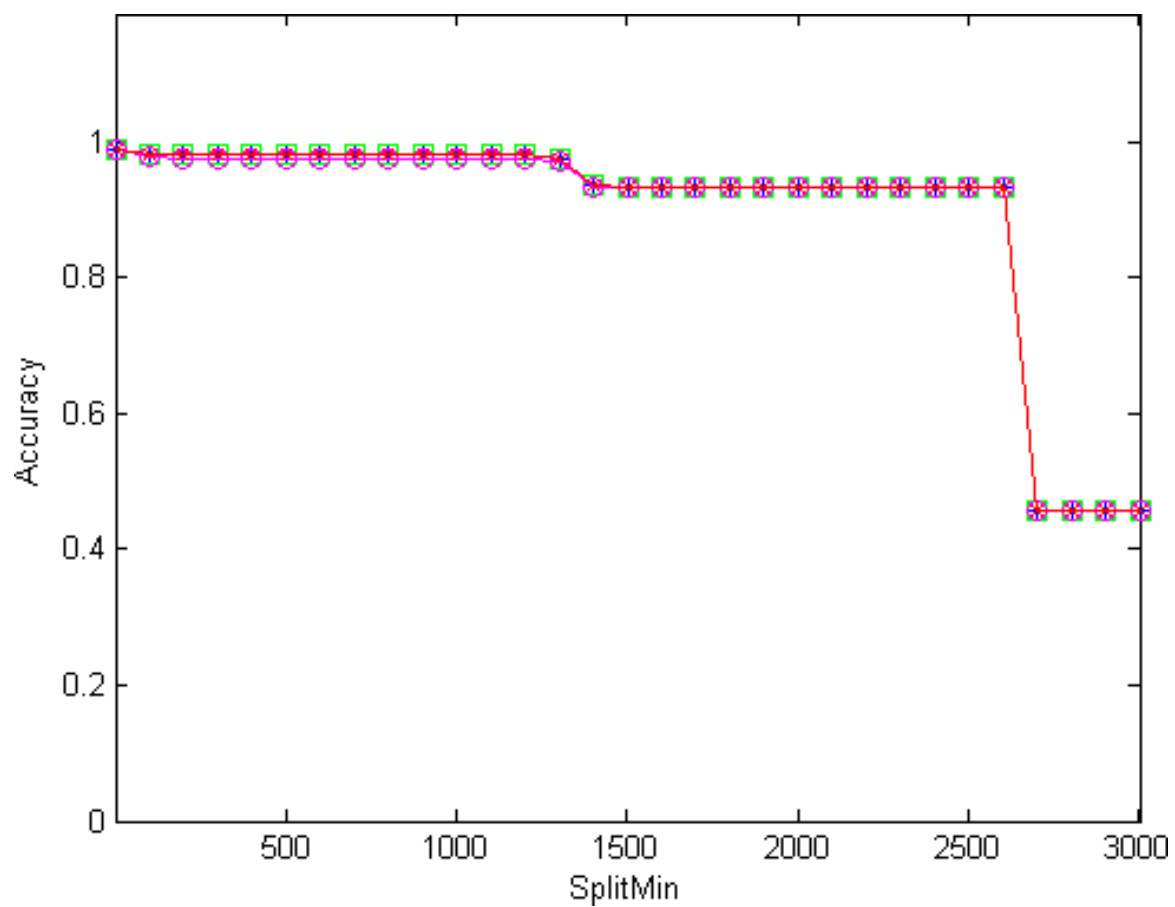

(b)
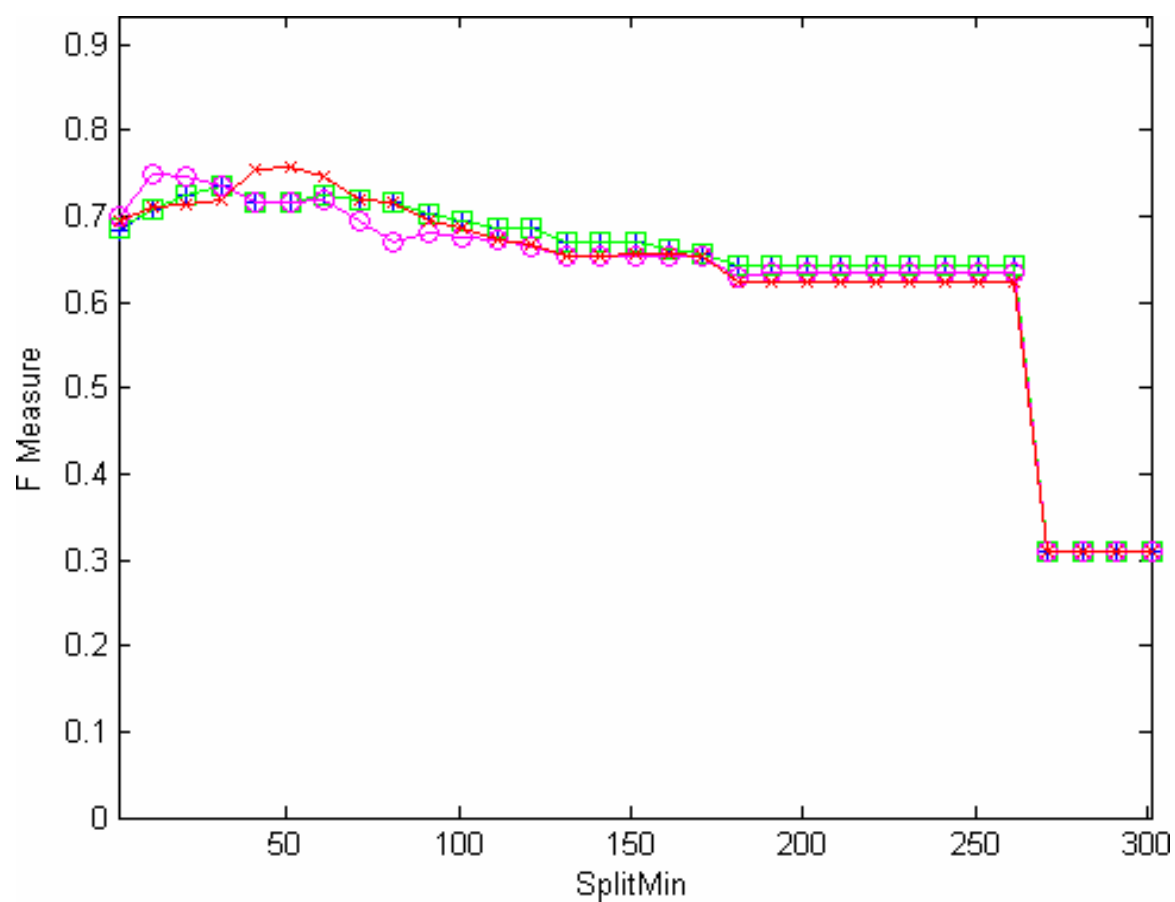
Figure 4.6 – Continued
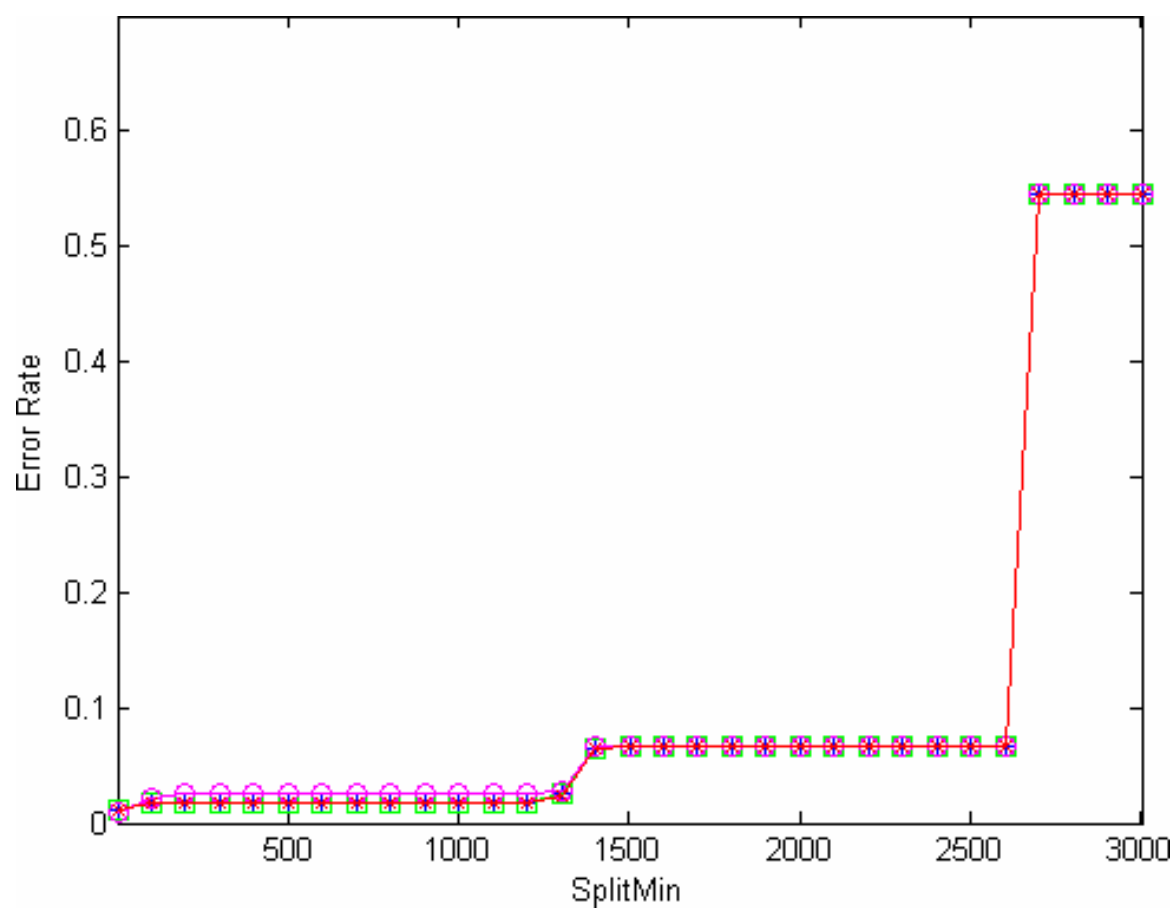
(c)

Figure 4.6 – Continued
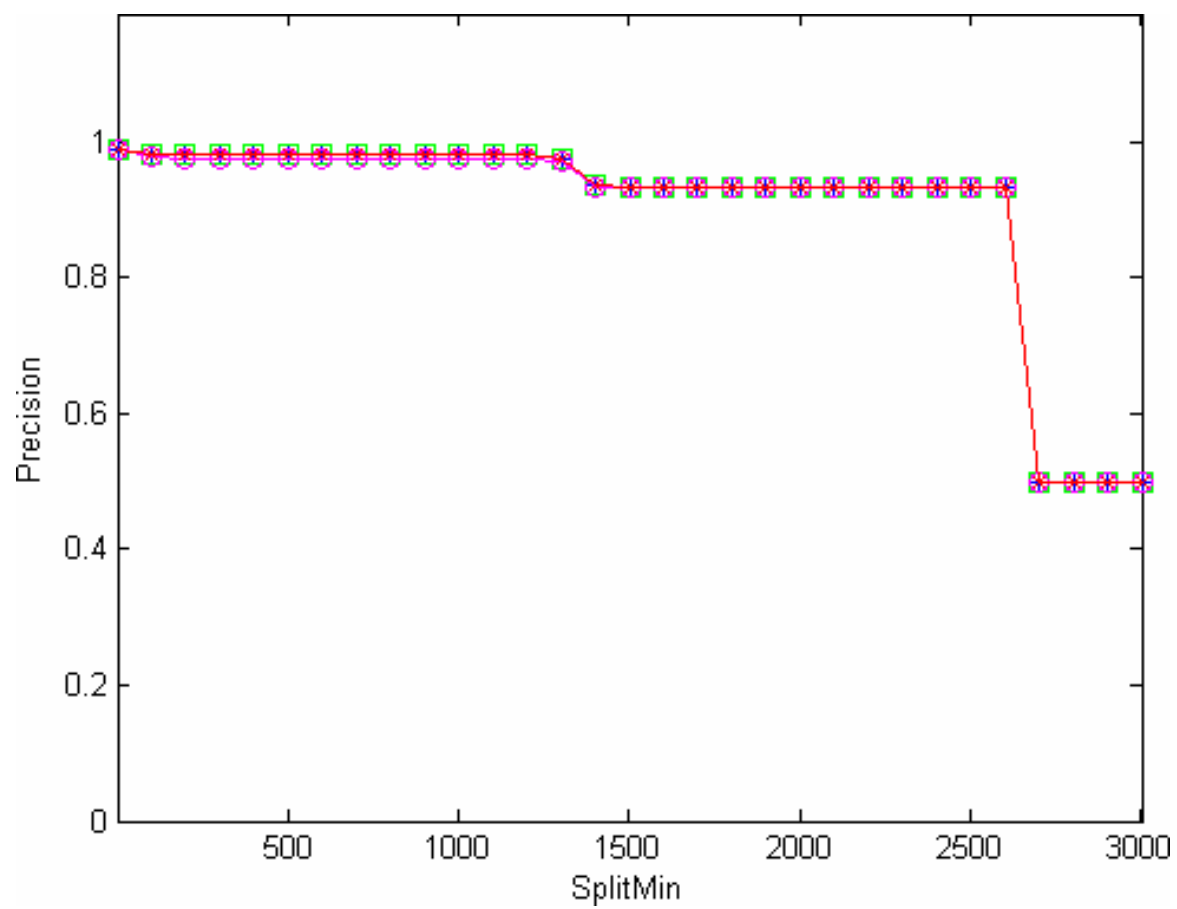

(d)

Figure 4.6 – Continued
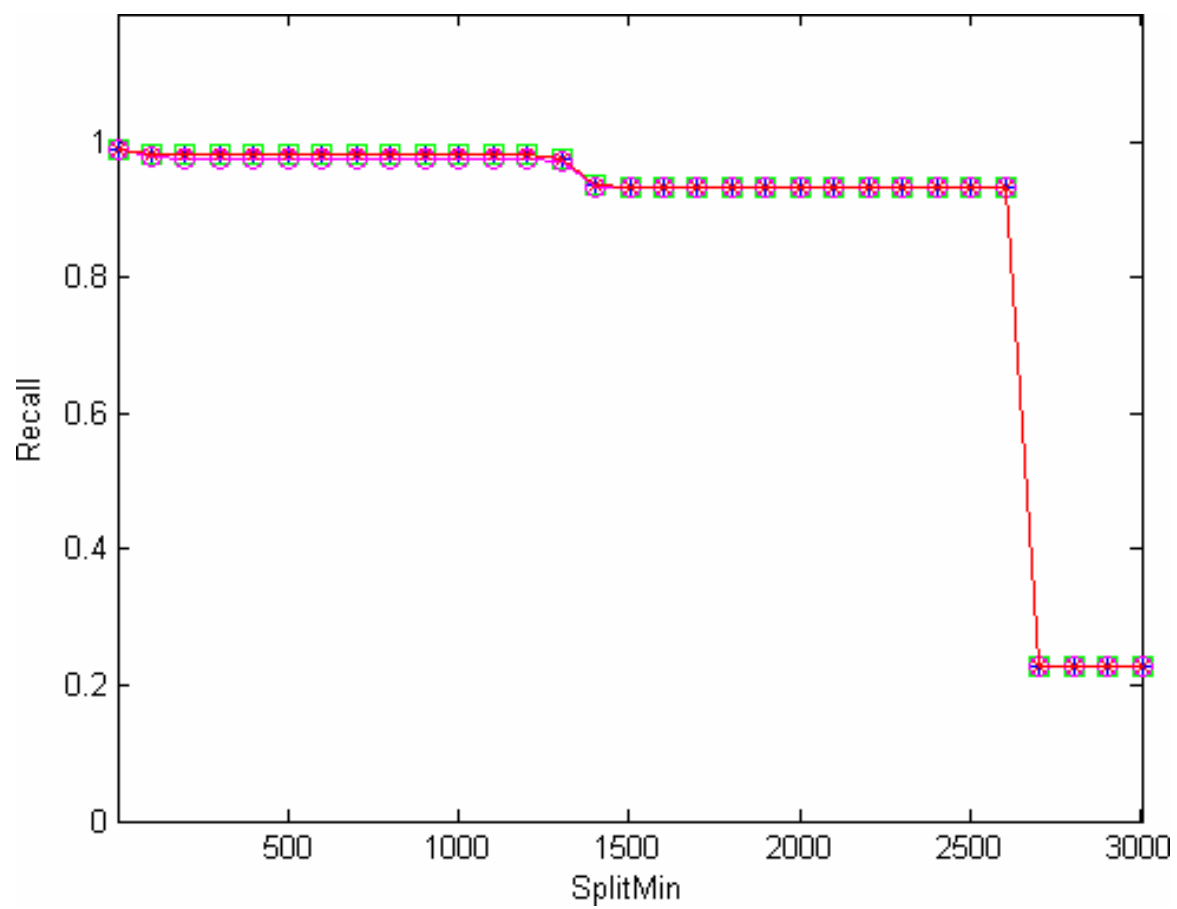


(e)

Figure 4.6 – Continued
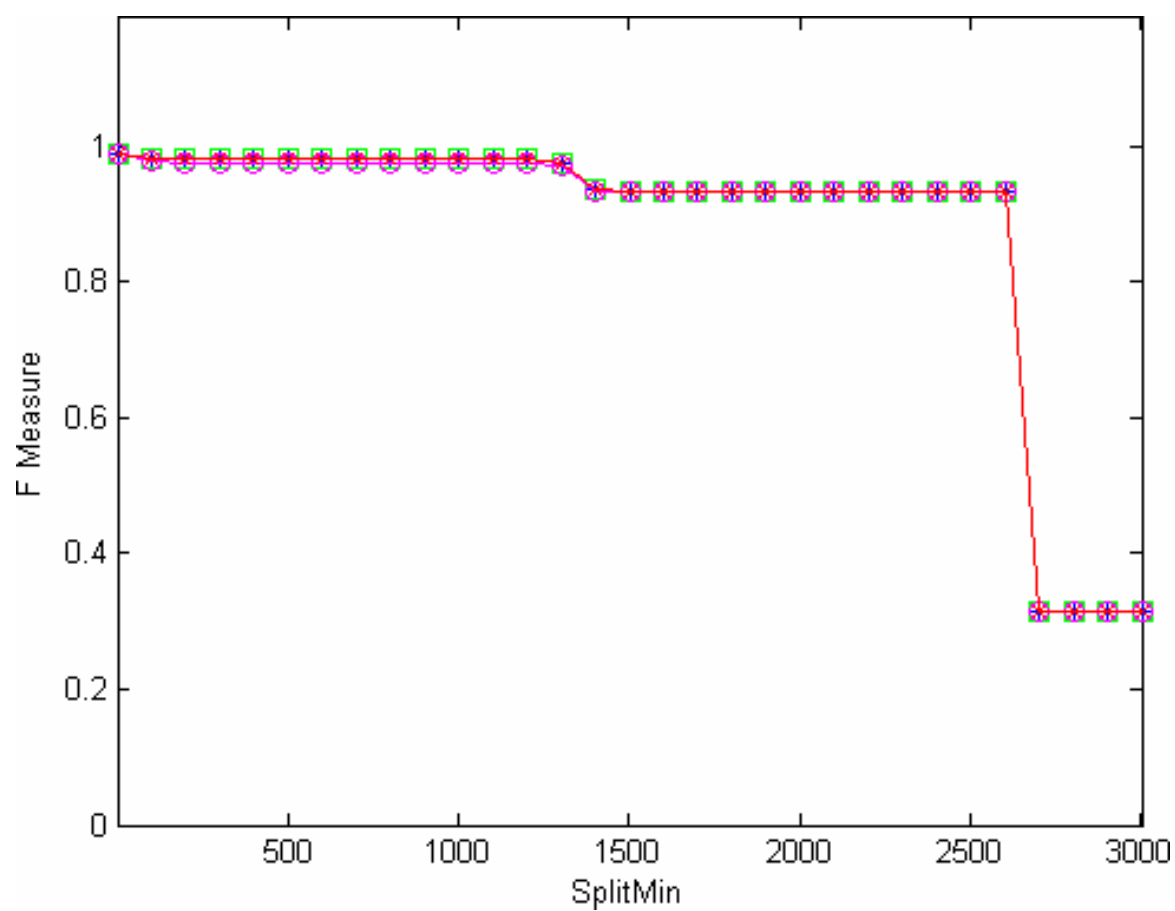

(f)

Figure 4.6 – Continued

(g)

Figure 4.6 – Continued
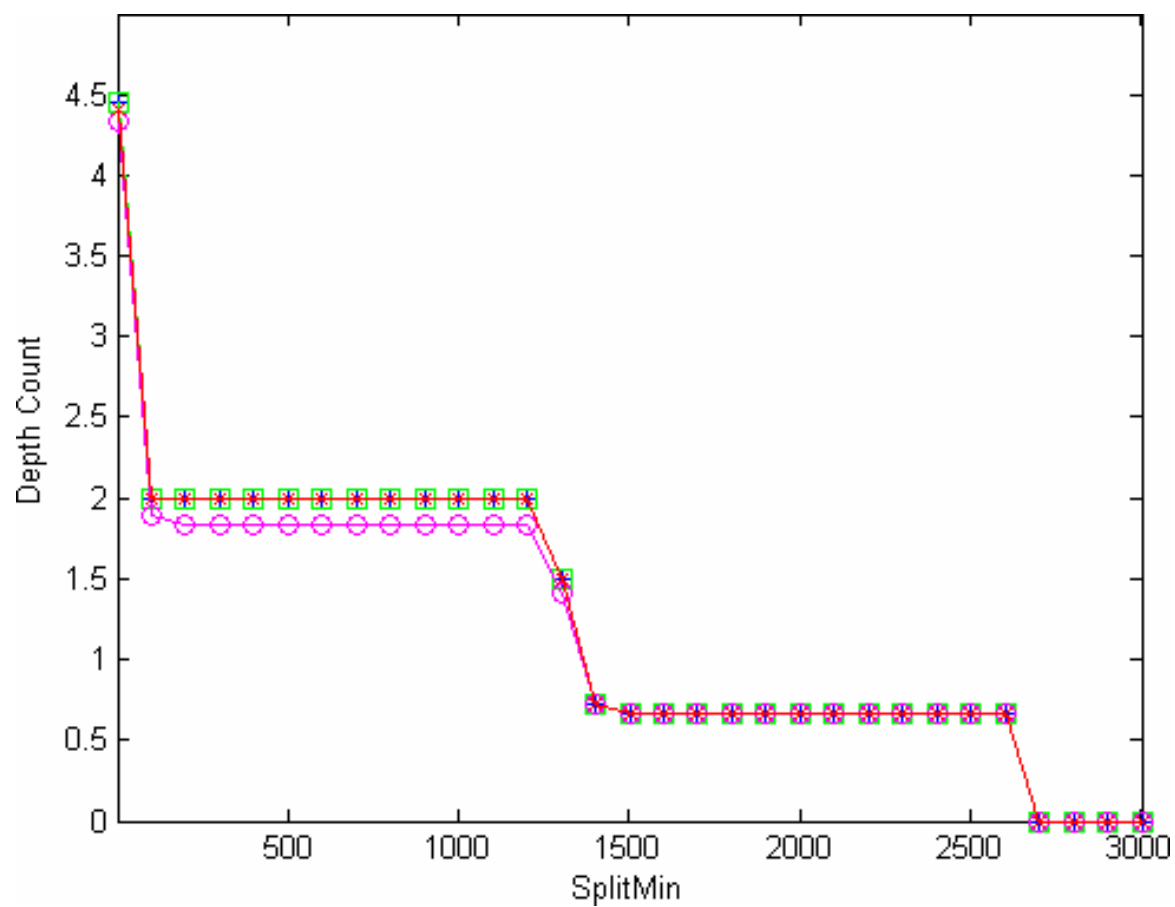
(h)

Figure 4.6 – Continued
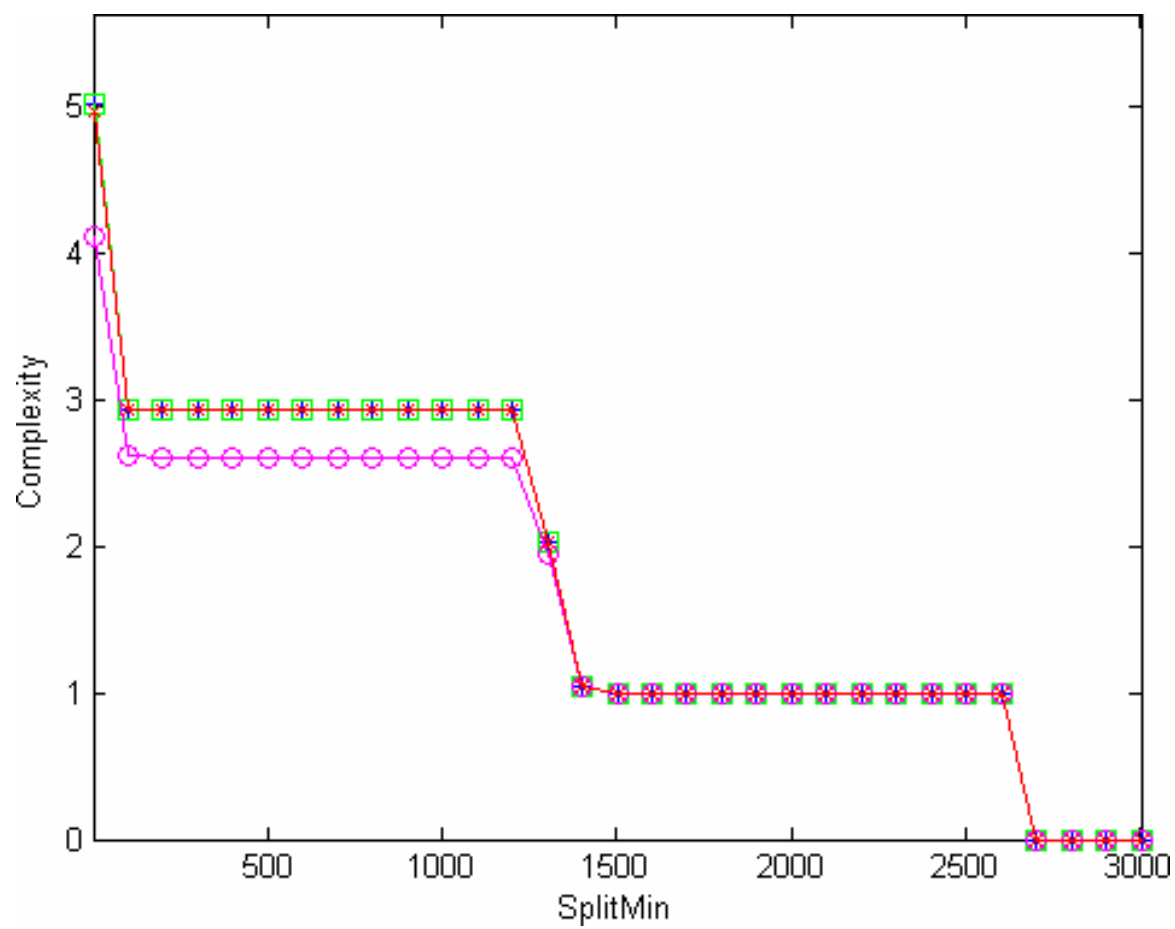
(i)

4.2.1.4 <u>Gaussian Experiment 3a – Moderately Noisy Case with Low Density</u>

This experiment is the first reasonably challenging test on the splitting criteria. We position the Gaussian distribution clusters closer to each other, such that they significantly overlap. The blue distribution contains 150 points that belong to $N(5,1)$, whereas the red distribution contains 150 points that belong to $N(6,1)$, as depicted in Figure 4.7(a). While there is more noise between the clusters, we still observe definite areas of high concentration in each distribution. Our interest firstly lies in whether the splitting criteria can successfully separate the relevant areas of the data set from the noisy areas of the data set. Secondly, we are interested in the similarities and differences between the classification and structural metric results.

The classification metrics for each splitting criterion, despite having more variance per splitmin, are similar, as shown in Figure 4.7(b, c, d, e, f). The Gini Index and Twoing Rule both on average perform better than the other criteria across all splitmins. An interesting observation however is the Rough Product has the best predictions overall of any splitting criterion between splitmins 41 and 61. In addition, the Rough Product generally tends to be the simplest criterion in a structural sense, as shown in Figure 4.7(g, h, i).

4.2.1.5 <u>Gaussian Experiment 3b – Moderately Noisy Case with High Density</u>

As in Gaussian Experiment 3a, the Gaussian distribution clusters remain close to each other and significantly overlap. However, we increase the density of each distribution cluster by an order of magnitude, such that each class contains 1500 data points. The blue distribution contains points that belong to $N(5,1)$, whereas the red 2b,

81
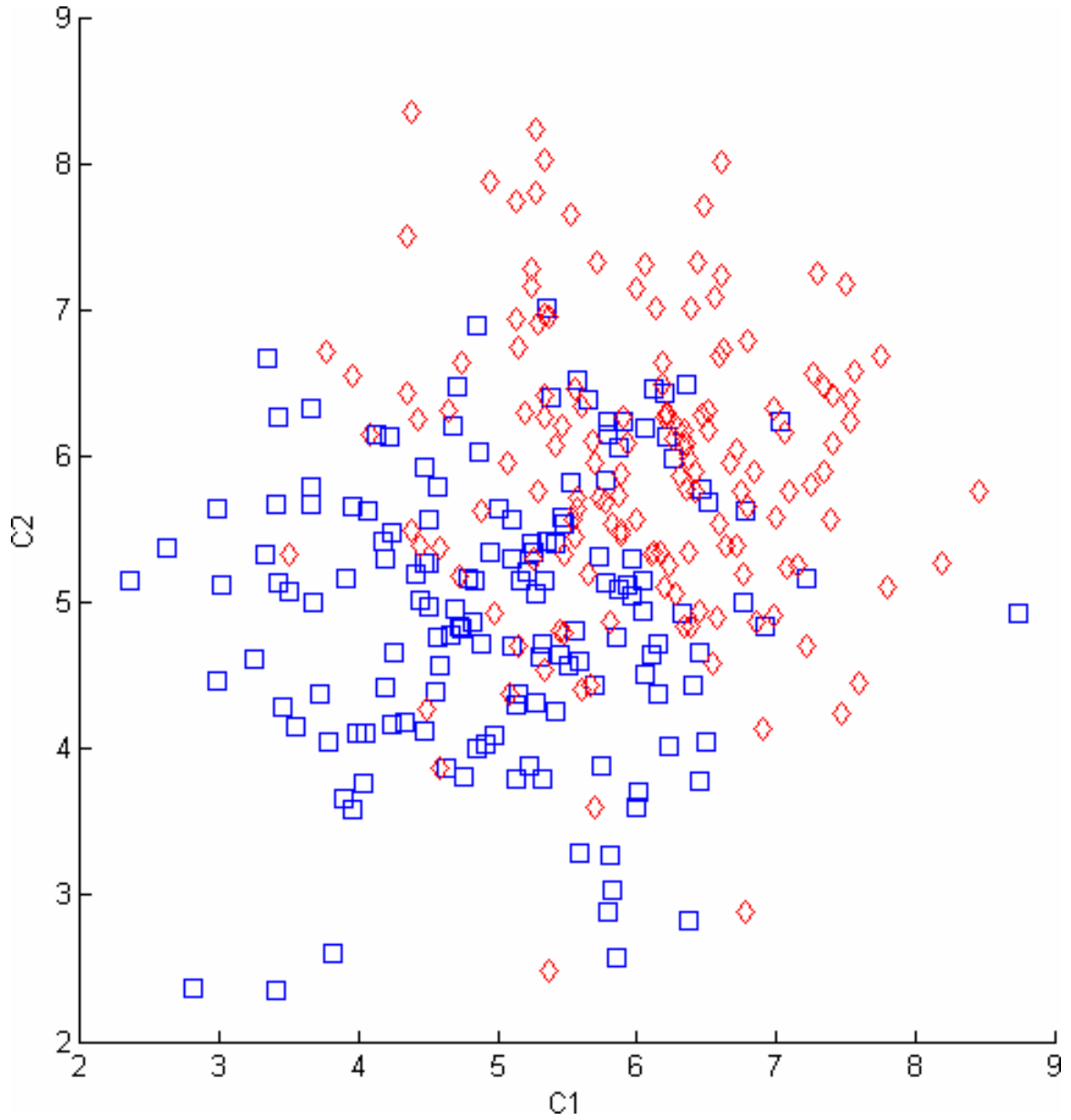
(a)

Figure 4.7: Gaussian Experiment 3a. The data set (a) is constructed such that 150 blue three-dimensional points come from $N(5,1)$ and 150 red three-dimensional points come from $N(6,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.7 – Continued
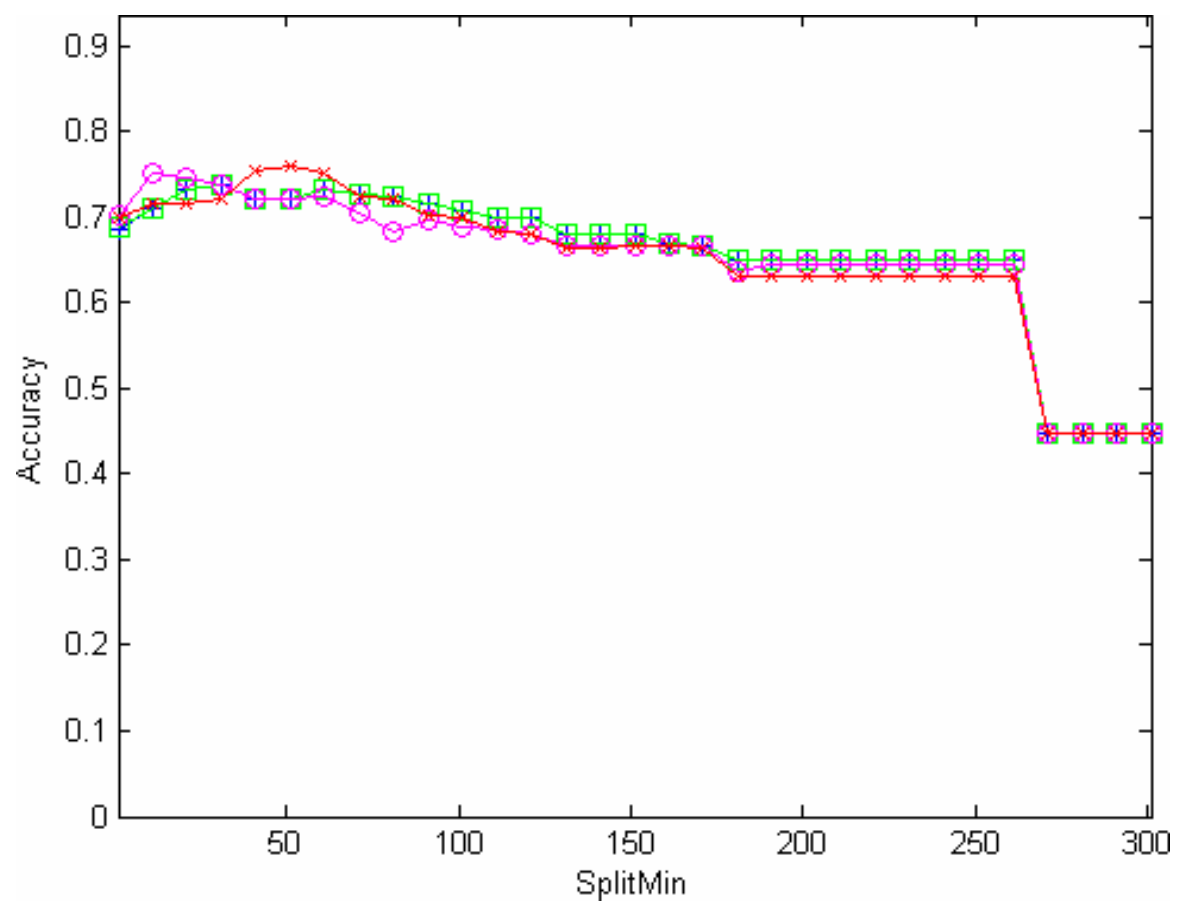
(b)

Figure 4.7 – Continued
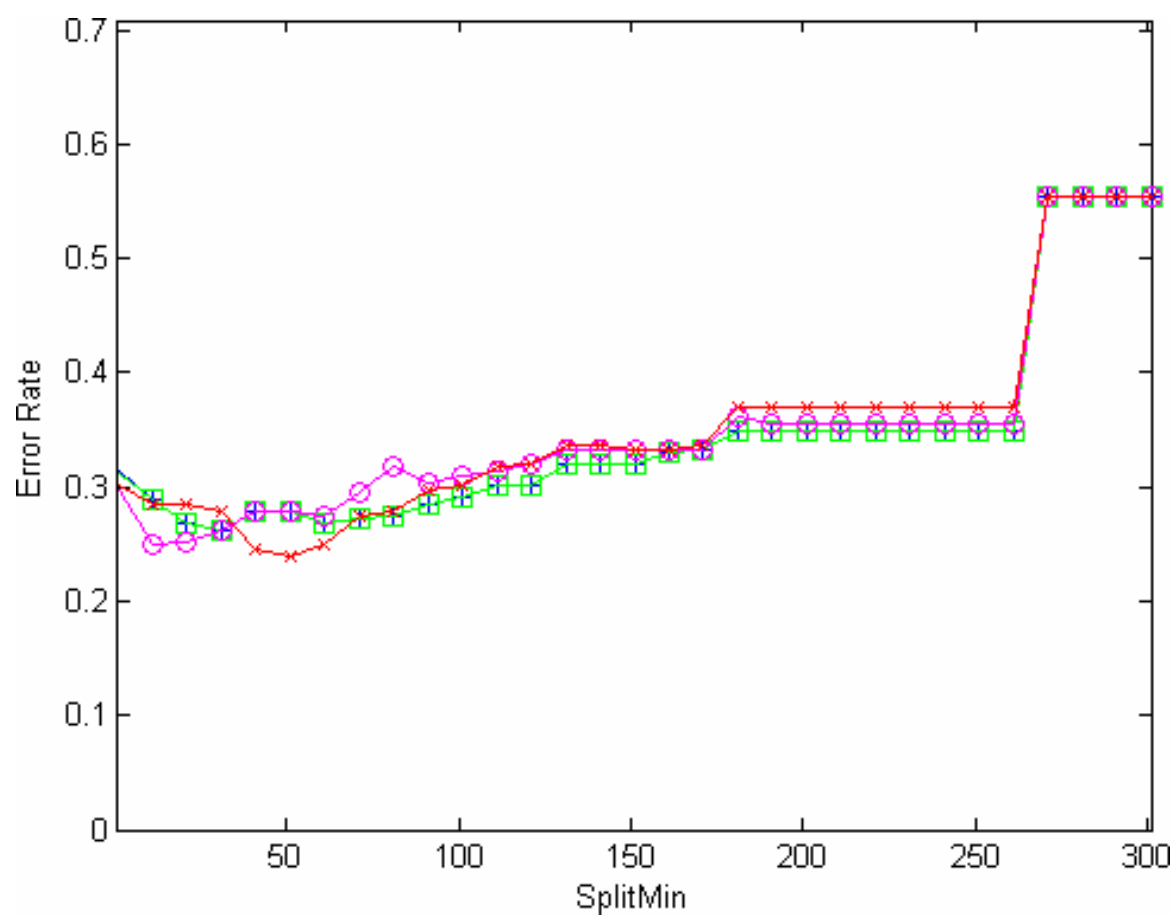
(c)

Figure 4.7 – Continued
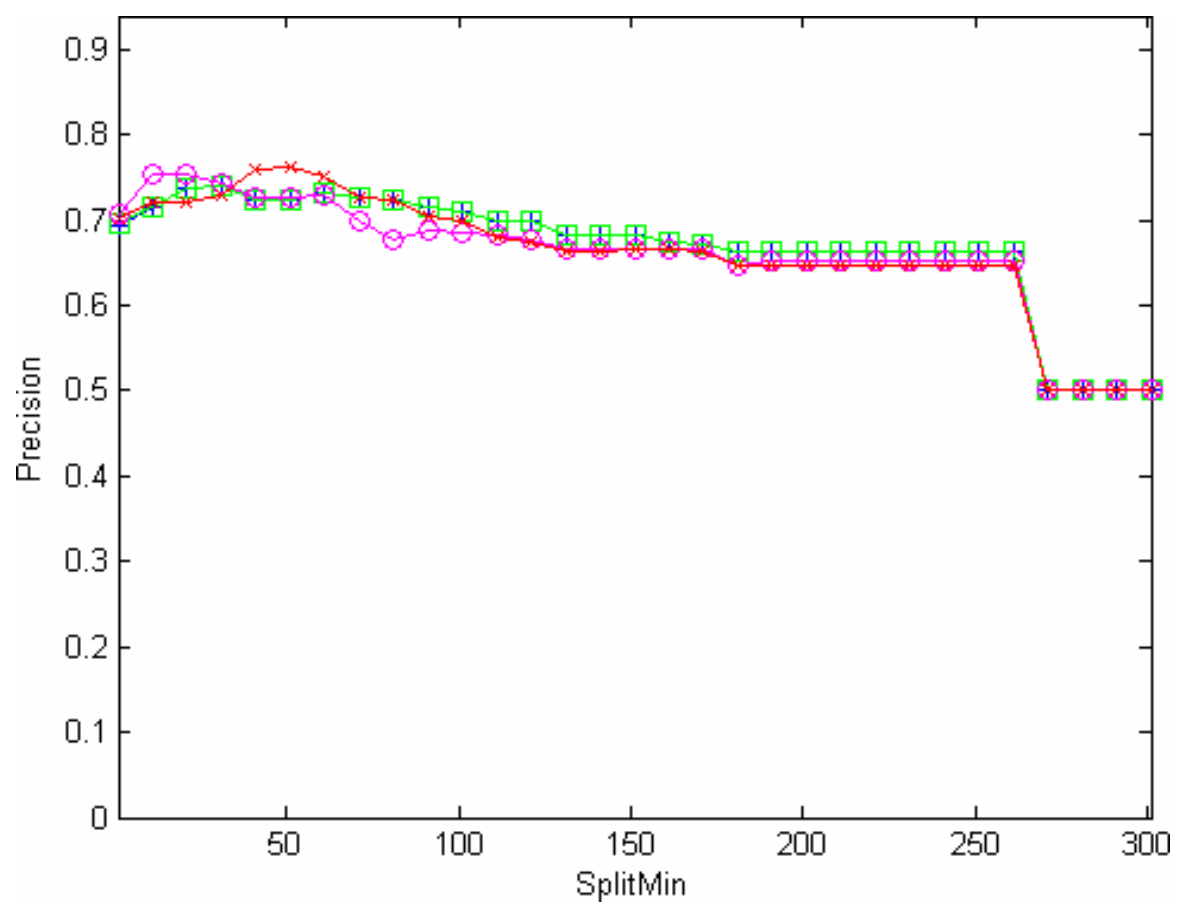
(d)

Figure 4.7 – Continued
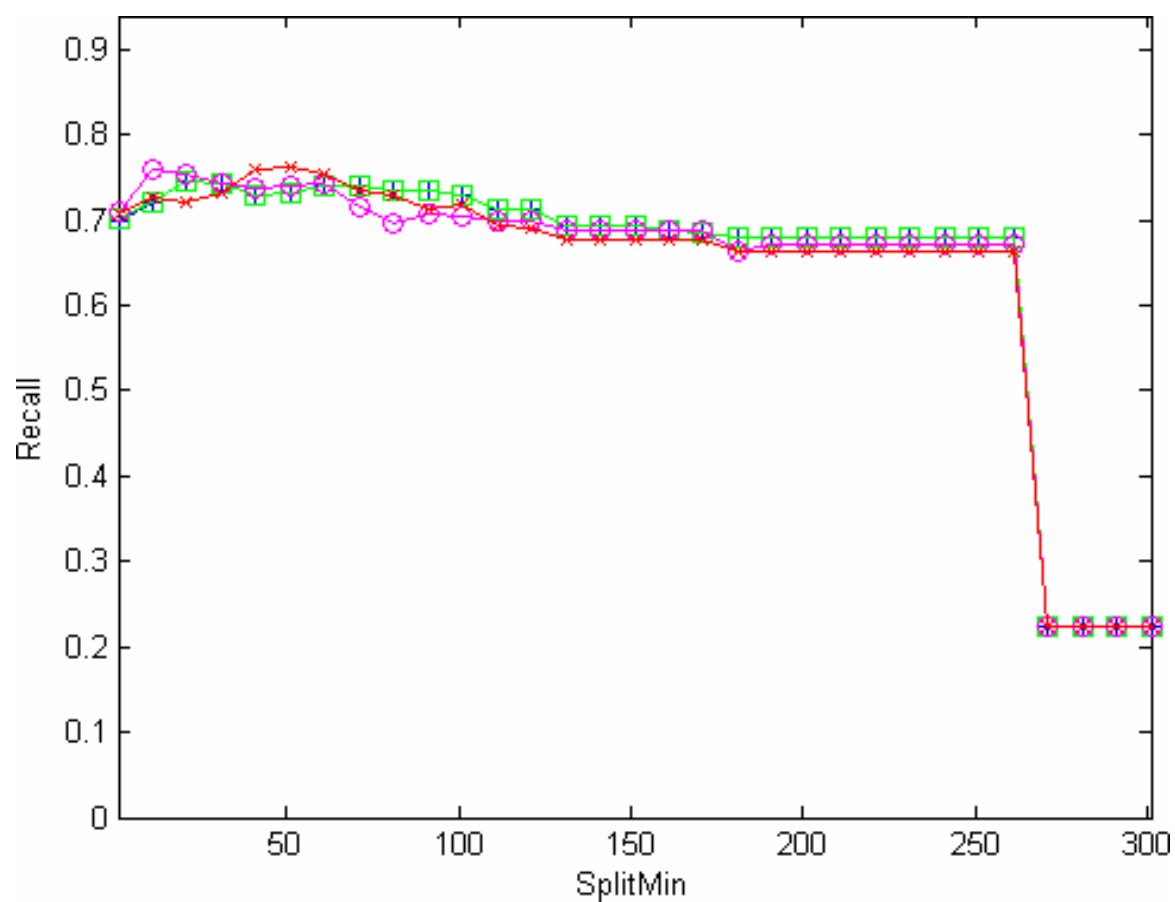
(e)

Figure 4.7 – Continued

(f)

Figure 4.7 – Continued
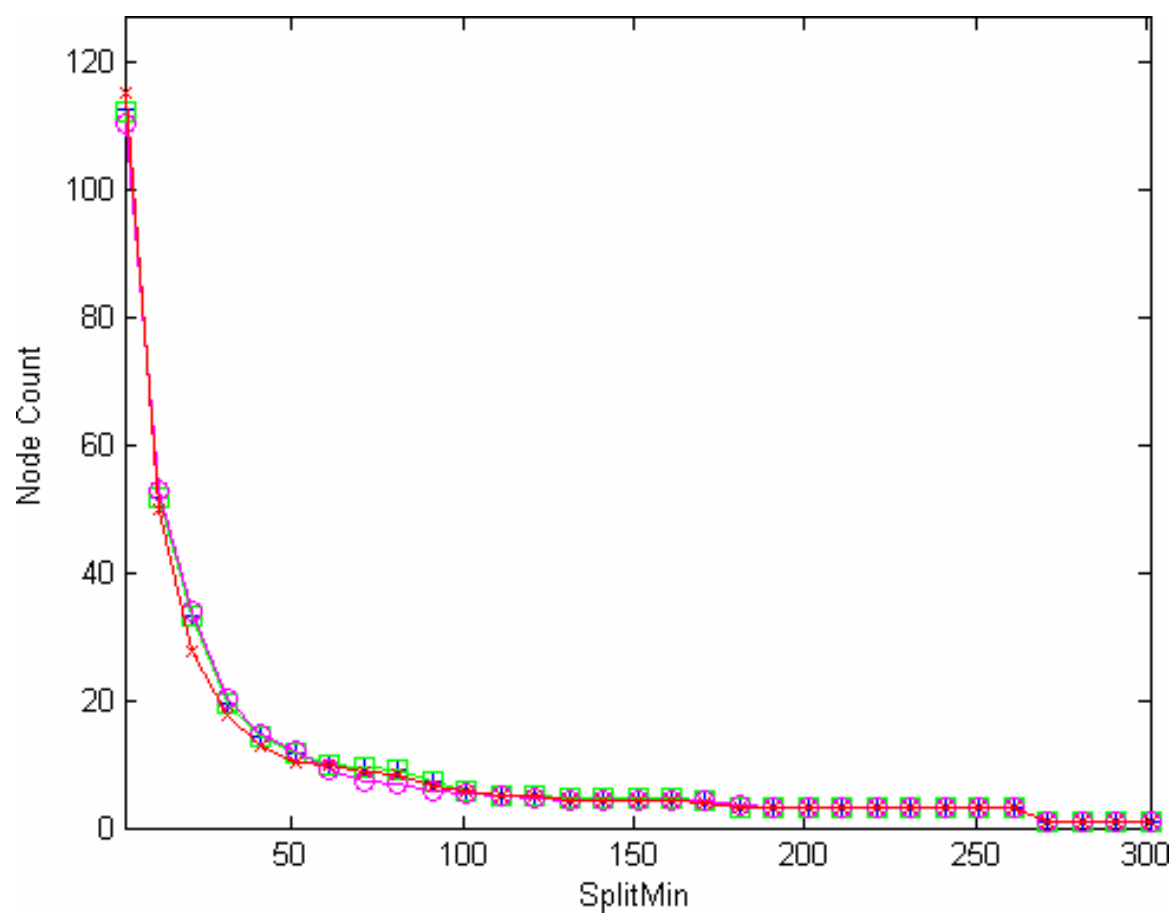
(g)

Figure 4.7 – Continued
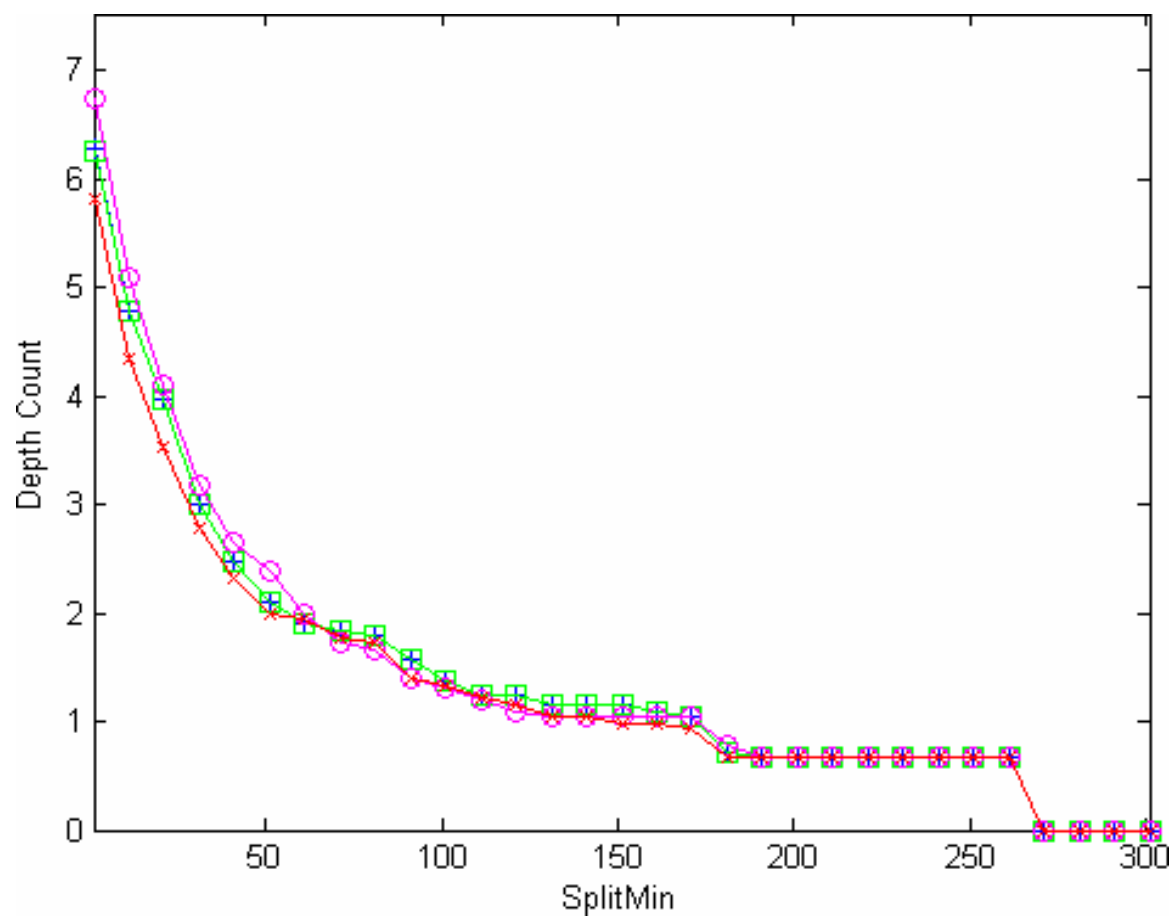
(h)

Figure 4.7 – Continued
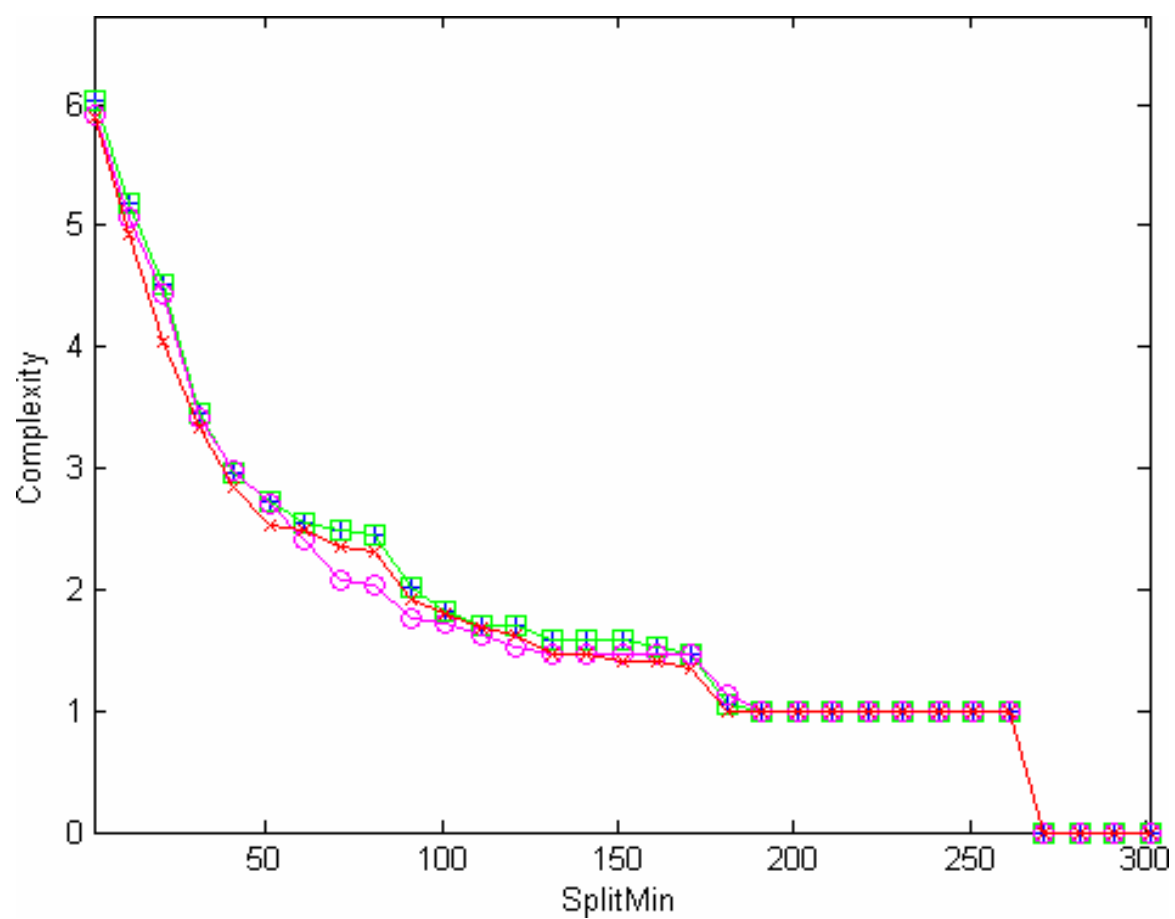
(i)

distribution contains points that belong to $N(6,1)$, as shown in Figure 4.8(a). We aim to determine whether a noticeable difference exists between the results in this experiment and those in Gaussian Experiment 3a. Based on the results from Experiments 2a and we expect the resultant decision trees in this experiment to be very similar to those produced in Gaussian Experiment 3a.

The classification metric results show very little variance between the splitting criteria at each splitmin. This may be due to the increased density in each distribution. The higher scatter in Experiment 3a may have made it more difficult to model the decision classes accurately and as such, the results from those test sets may have been less stable. Despite this difficulty, the classification results of this experiment, as shown in Figure 4.8(b, c, d, e, f), are similar to those in Experiment 3a. For the structural metrics, shown in Figure 4.8(g, h, i), we observe an overall dramatic decrease in tree depth and complexity for all the splitting criterions when compared to the same results in Experiment 3a. Interestingly, this phenomenon did not occur in Experiment 2a and 2b. Nevertheless, the Rough Product produces the lowest mean tree depth and mean complexity across all folds.

4.2.1.6   Gaussian Experiment 4a – Non-Separable Case with Low Density

In this experiment, we position the Gaussian distribution clusters in the exact same location, such that each cluster is practically indistinguishable from the other. Each distribution cluster contains 150 points that belong to $N(5,1)$, as shown in Figure 4.9(a). We aim to determine the behavior of the splitting criteria in response to data that is intrinsically unclassifiable.
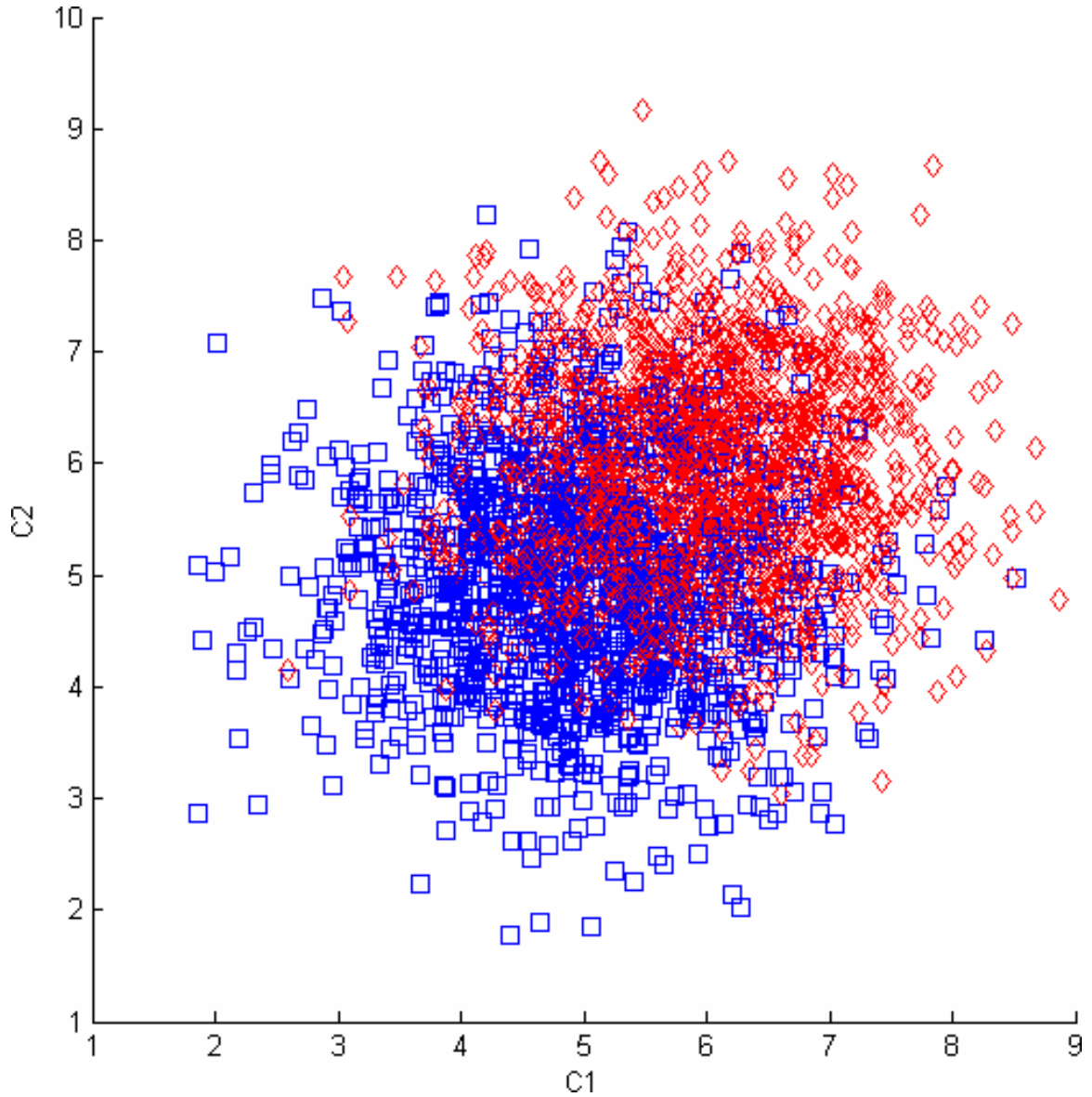
(a)

Figure 4.8: Gaussian Experiment 3b. The data set (a) is constructed such that 1500 blue three-dimensional points come from $N(5,1)$ and 1500 red three-dimensional points come from $N(6,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.8 – Continued
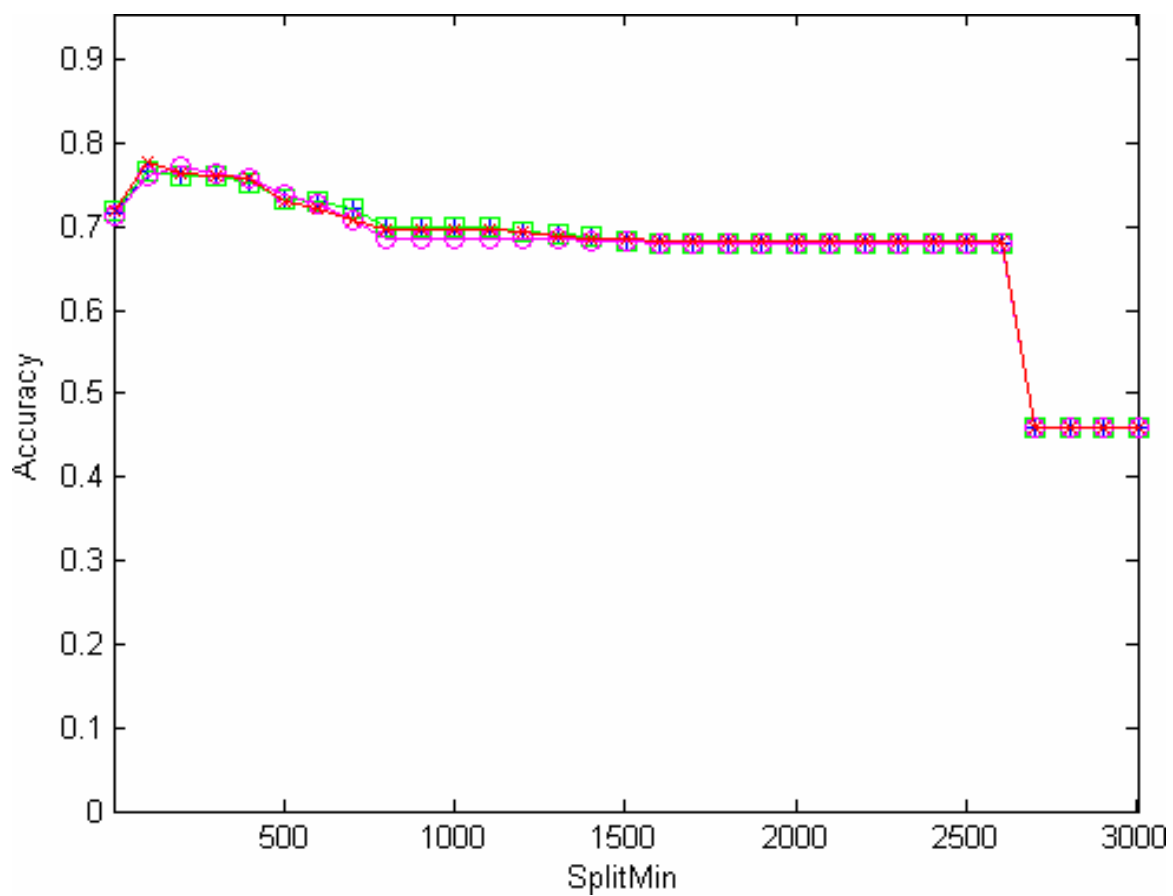
(b)

Figure 4.8 – Continued
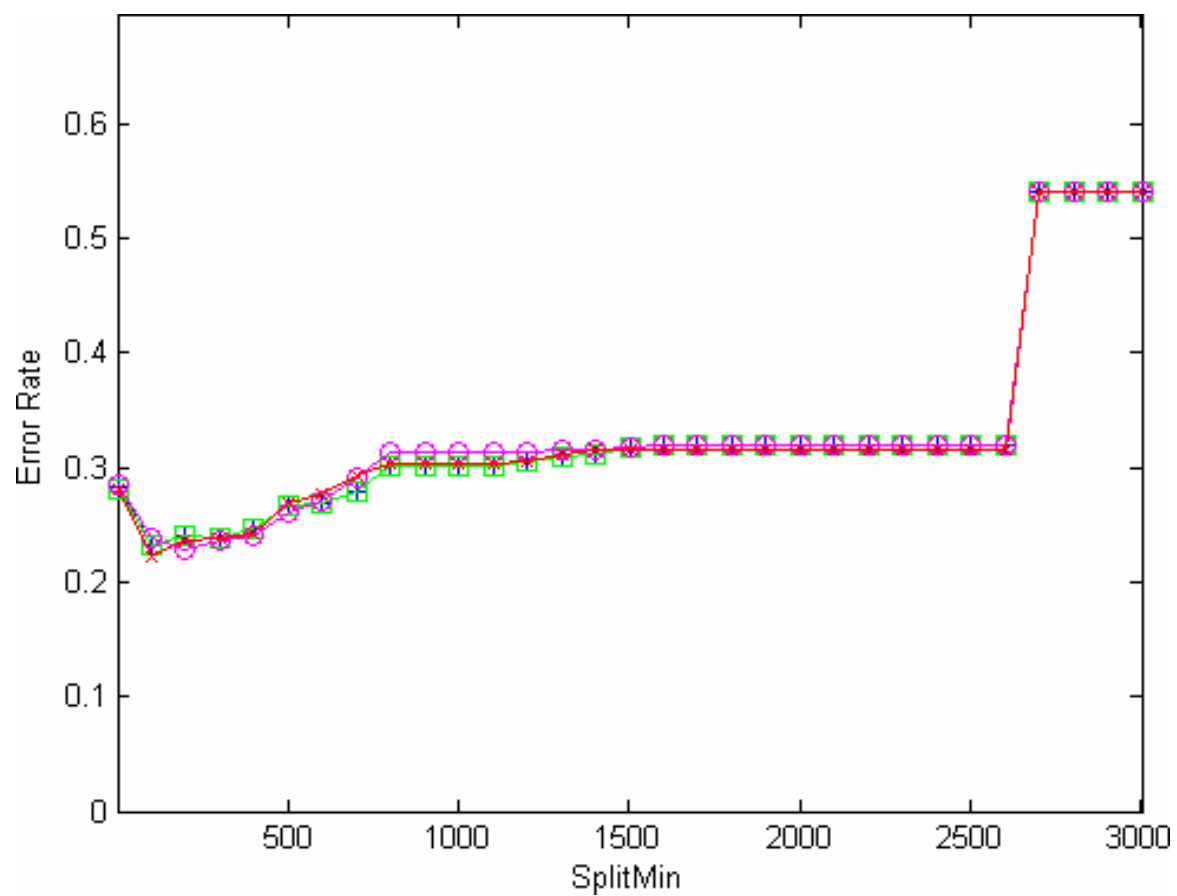
(c)

Figure 4.8 – Continued
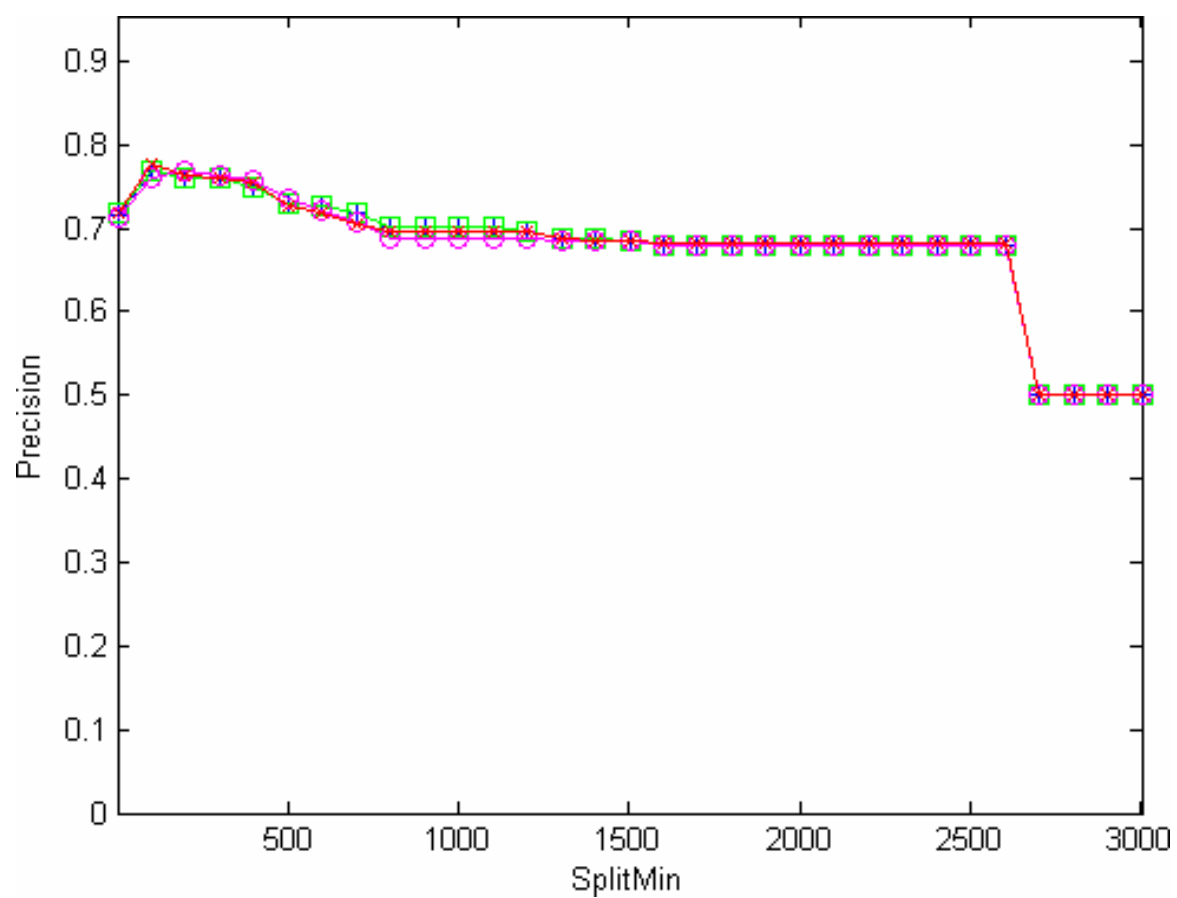
(d)

Figure 4.8 – Continued
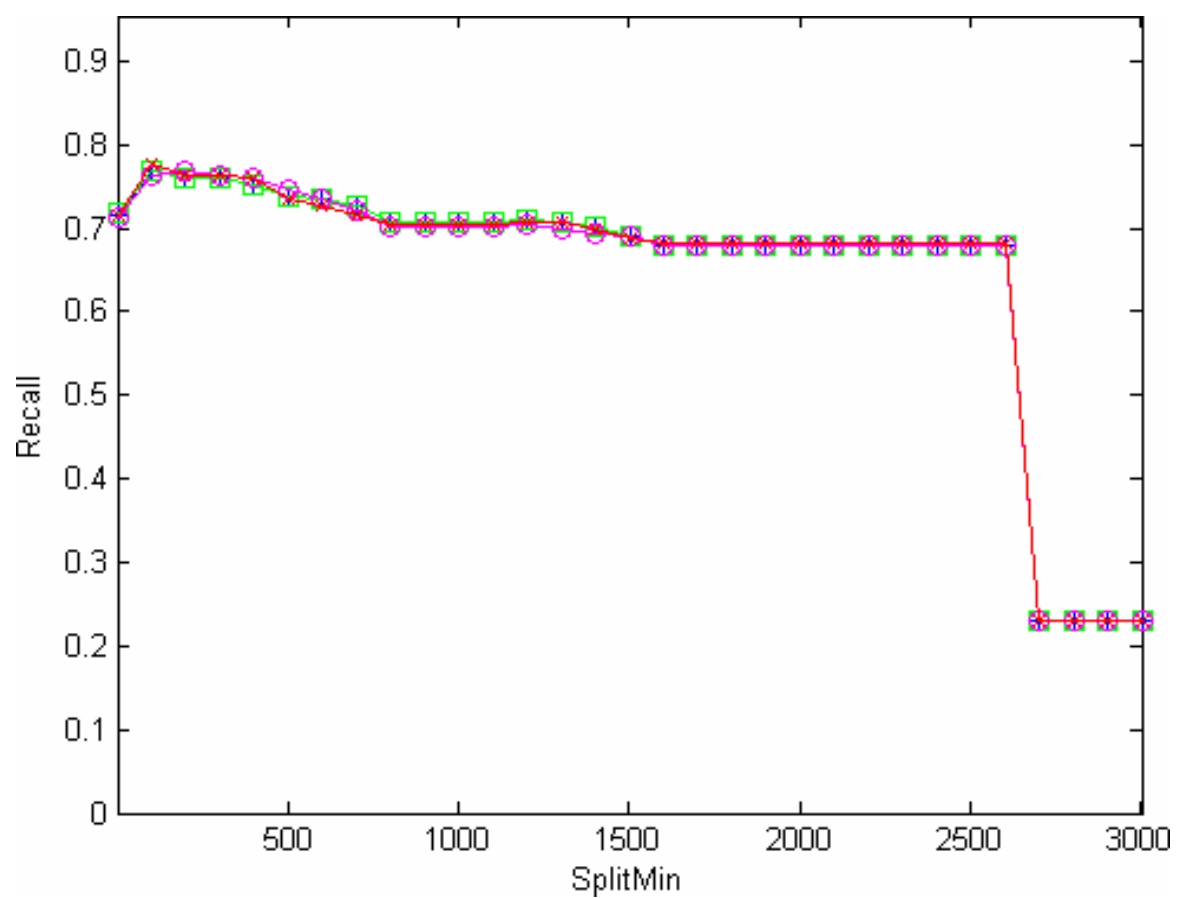
(e)

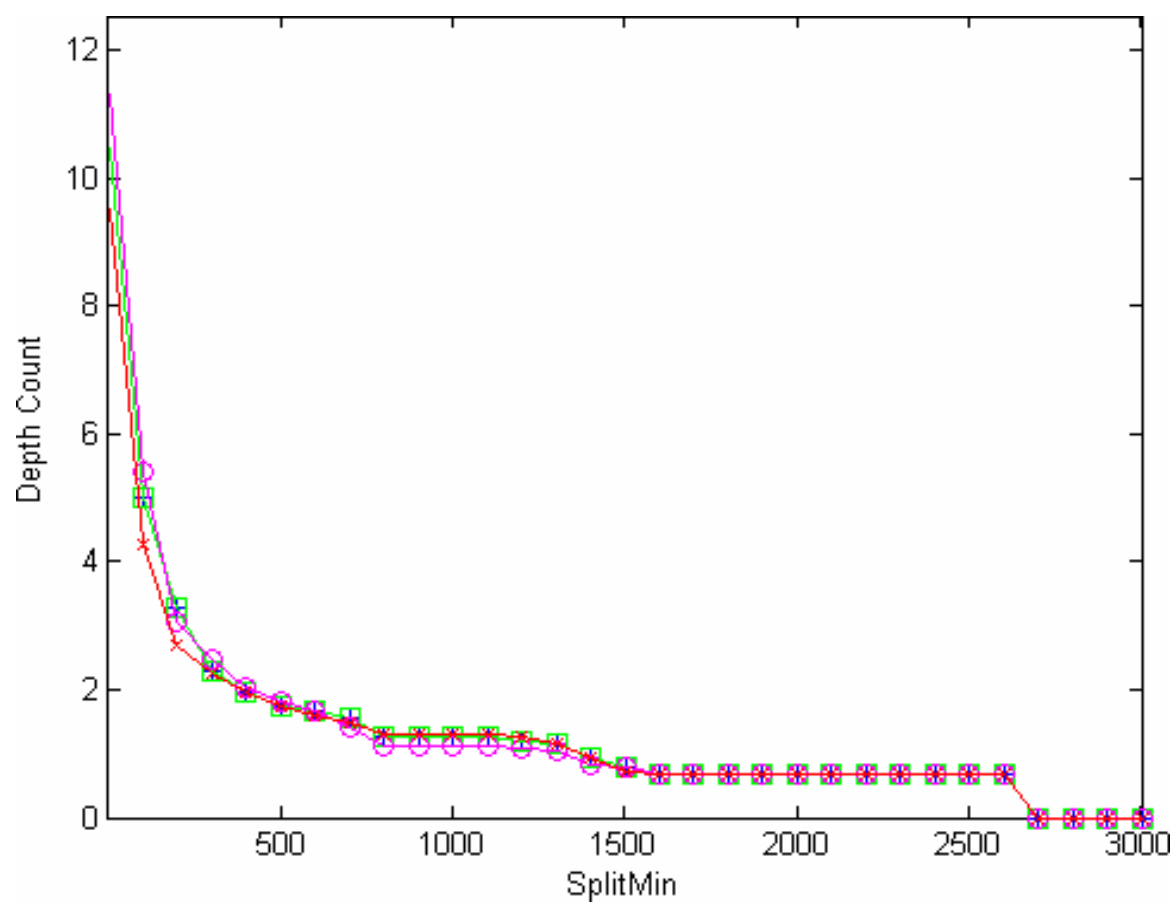Figure 4.8 – Continued
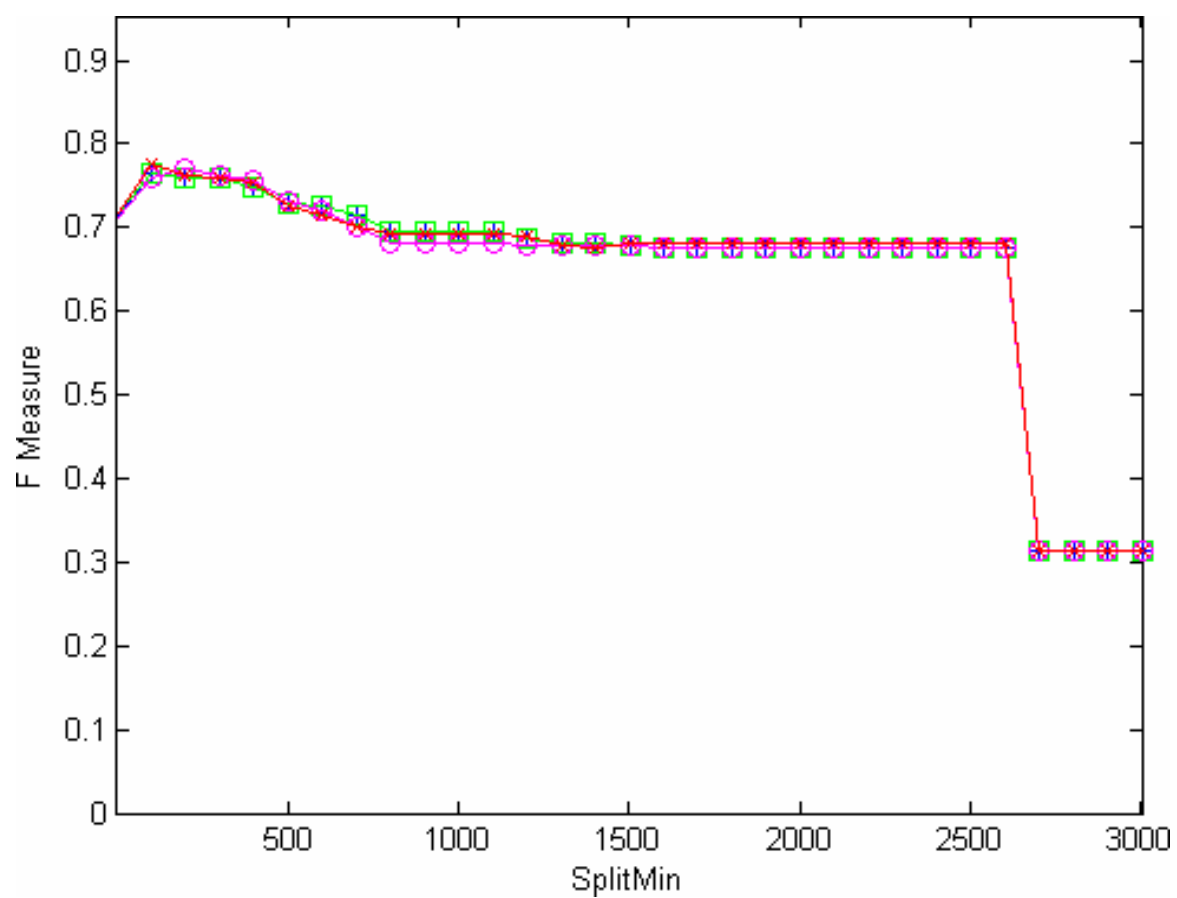
(f)

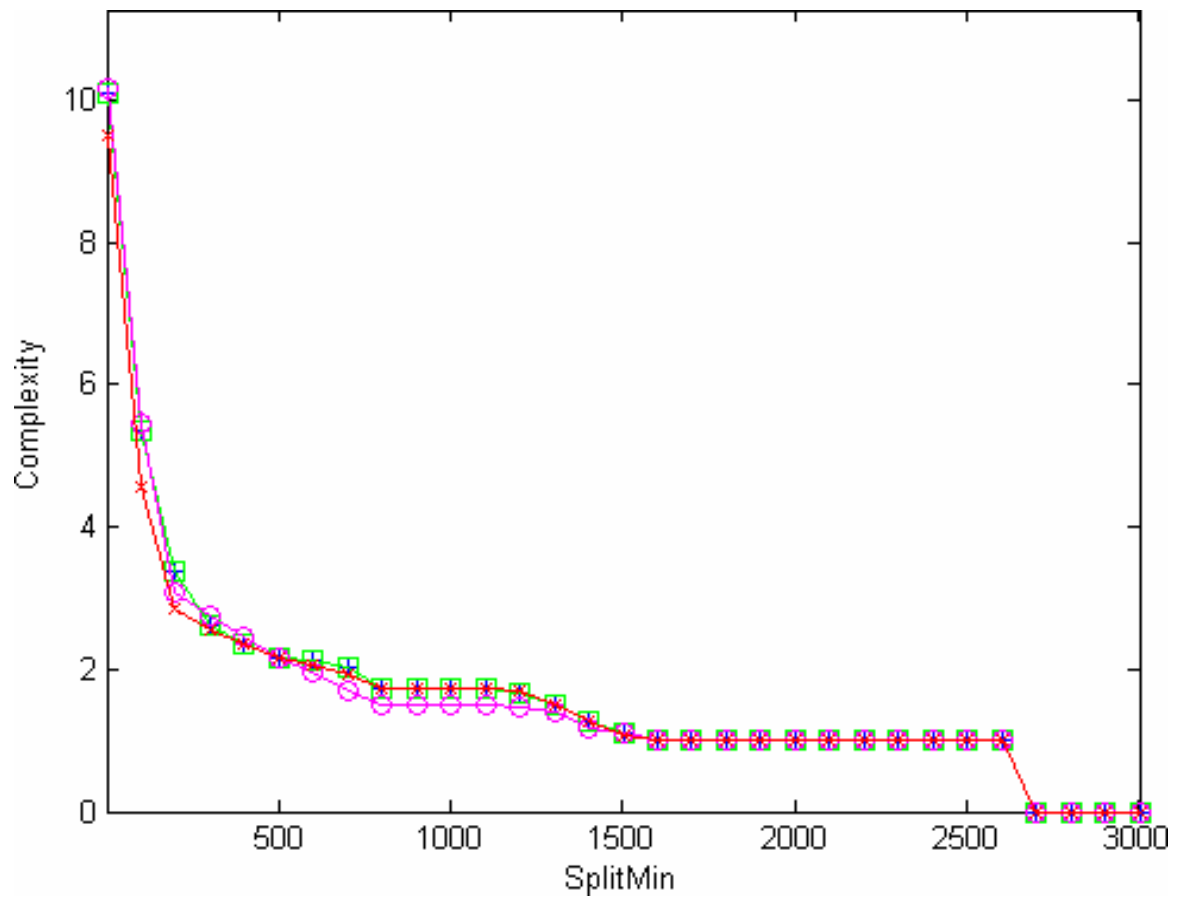Figure 4.8 – Continued
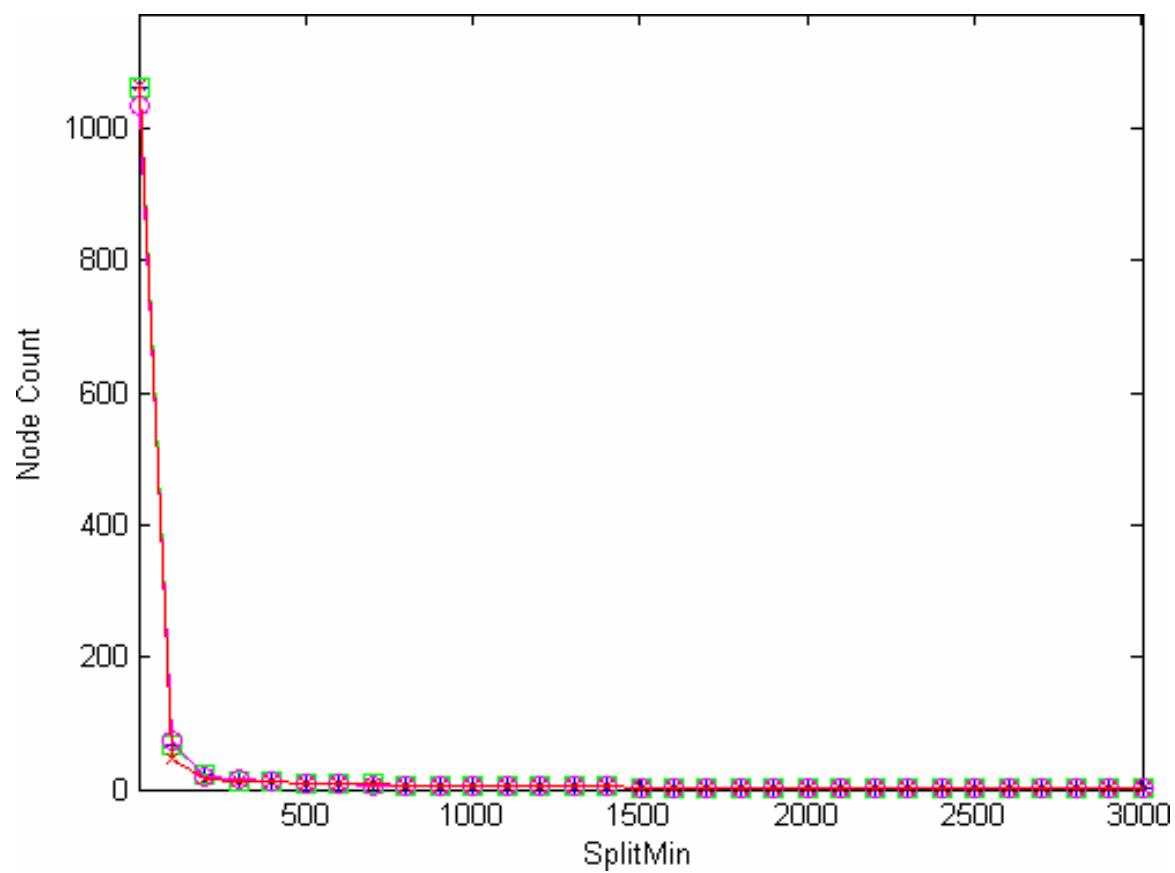
(g)

Figure 4.8 – Continued

(h)

Figure 4.8 – Continued

(i)

The classification metric results, shown in Figure 4.9 (b, c, d, e, f) show that the splitting criteria behave as expected. The criteria choose splits that do not produce significant classification improvements. However, despite this observation, the relatively steep gradient between splitmins 261 and 271 implies that initial splits at the root of the tree cause the greatest classification improvements, independent of the noise in the data. The structural metric results, shown in Figure 4.9(g, h, i), are surprising. The Rough Product splitting criterion node count, tree depth, and complexity are, on average, 25% to 50% lower than the corresponding results for the Gini Index, Twoing Rule, and MDR splitting criteria. This phenomenon exists in Experiments 3a and 3b, but is much more pronounced in this experiment. In the presence of high noise, the Rough Product splitting criterion appears to keep the structural complexity low.

4.2.1.7   Gaussian Experiment 4b – Non-Separable Case with High Density

As in Experiment 4a, we position two Gaussian distribution clusters in the exact same location, such that each cluster is practically indistinguishable from the other. However, we increase the density of each distribution by an order of magnitude, shown in Figure 4.10(a), such that each cluster contains 1500 data points. We aim to determine whether a noticeable difference exists between the results in this experiment and Gaussian Experiment 4a, regarding the structural metric results with particular interest.

The classification metric results, as shown in Figure 4.10 (b, c, d, e, f) are consistent with those in Experiment 4a. In addition, the variance between the splitting

101
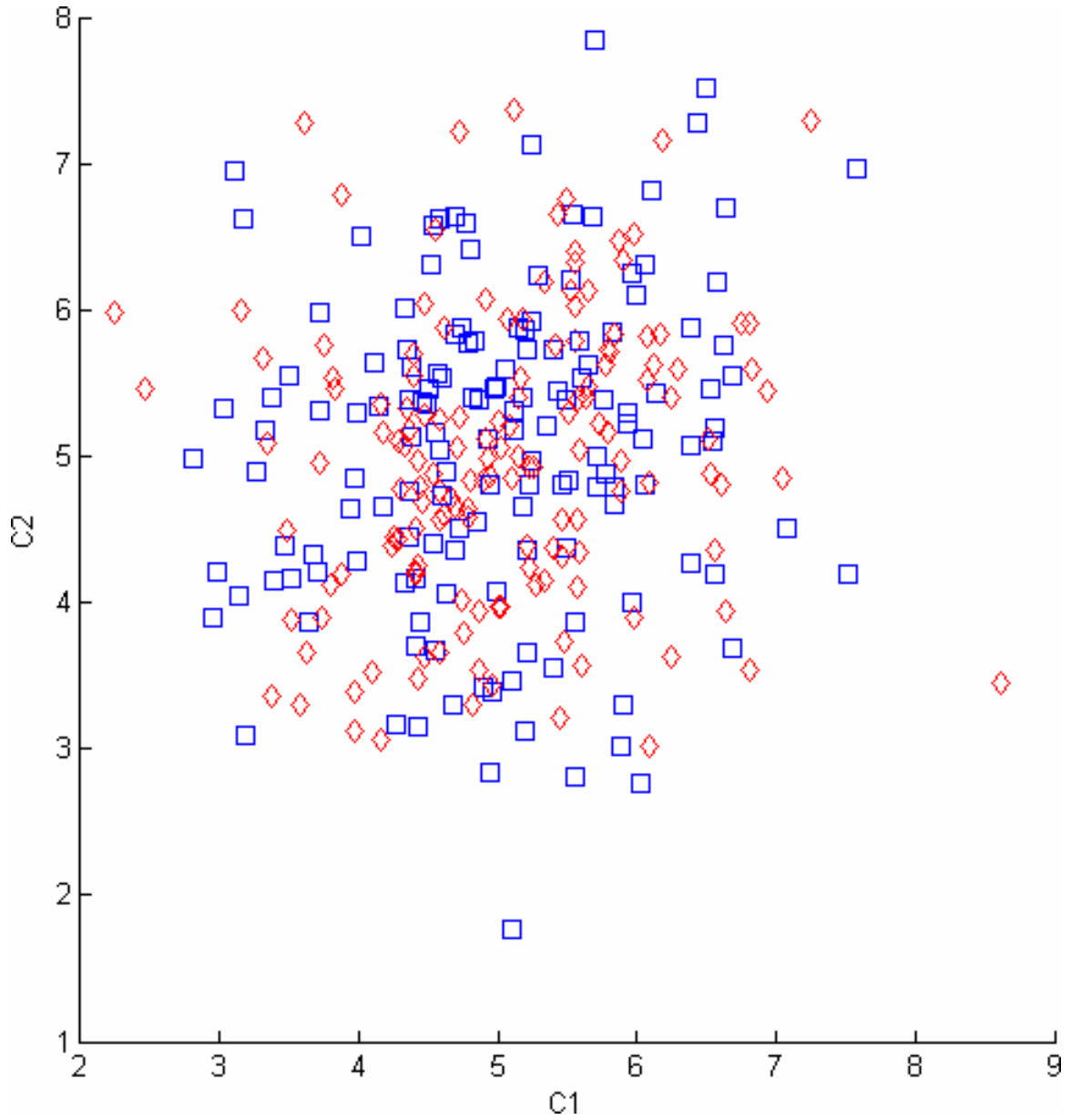
(a)

Figure 4.9: Gaussian Experiment 4a. The data set (a) is constructed such that 150 blue three-dimensional points come from $N(5,1)$ and 150 red three-dimensional points come from $N(5,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)
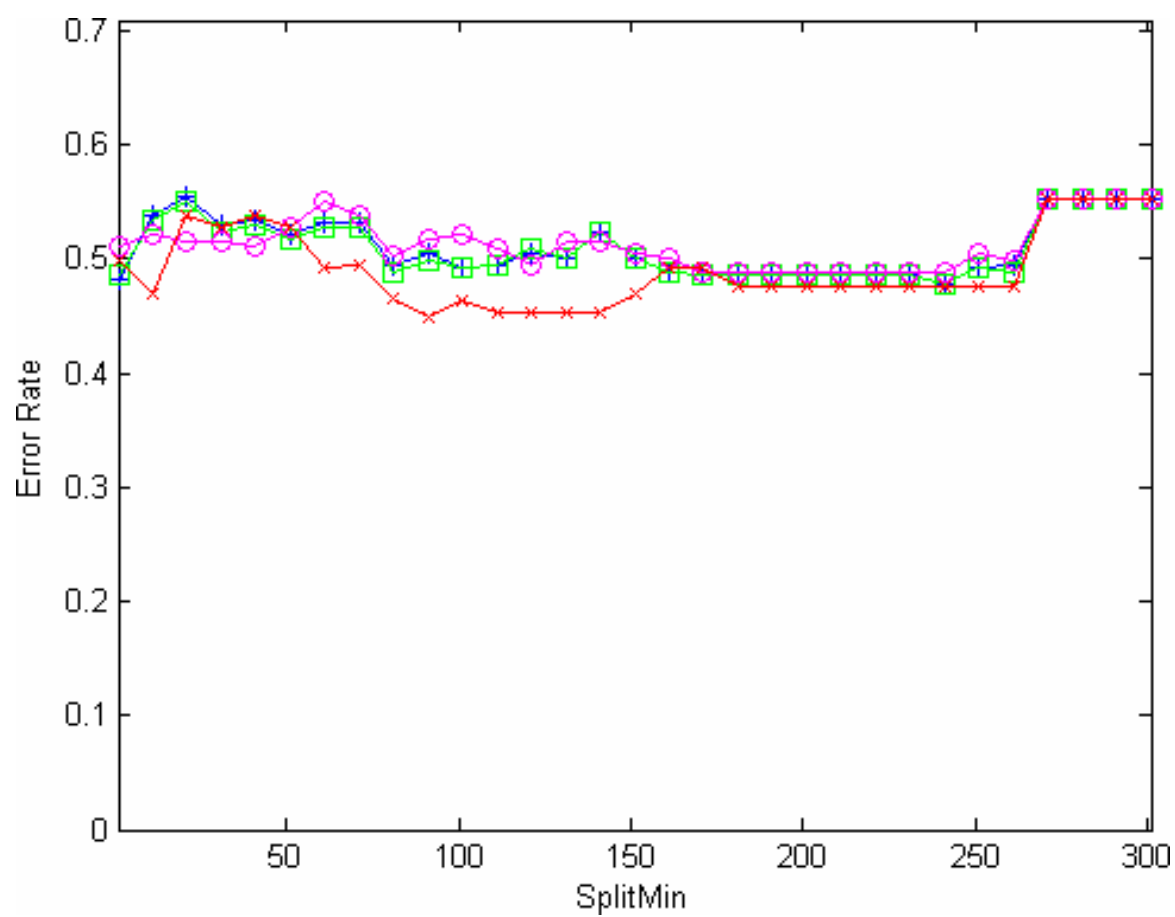
Figure 4.9 – Continued
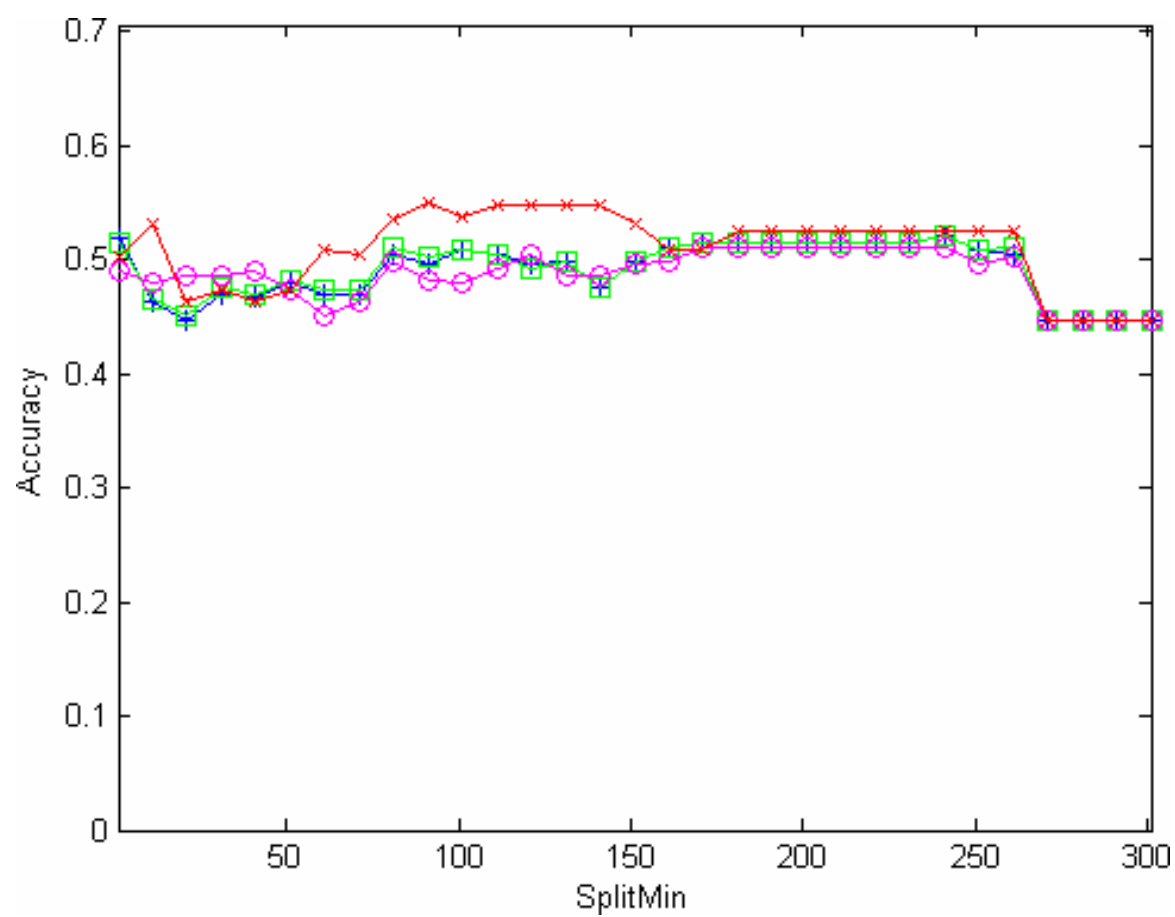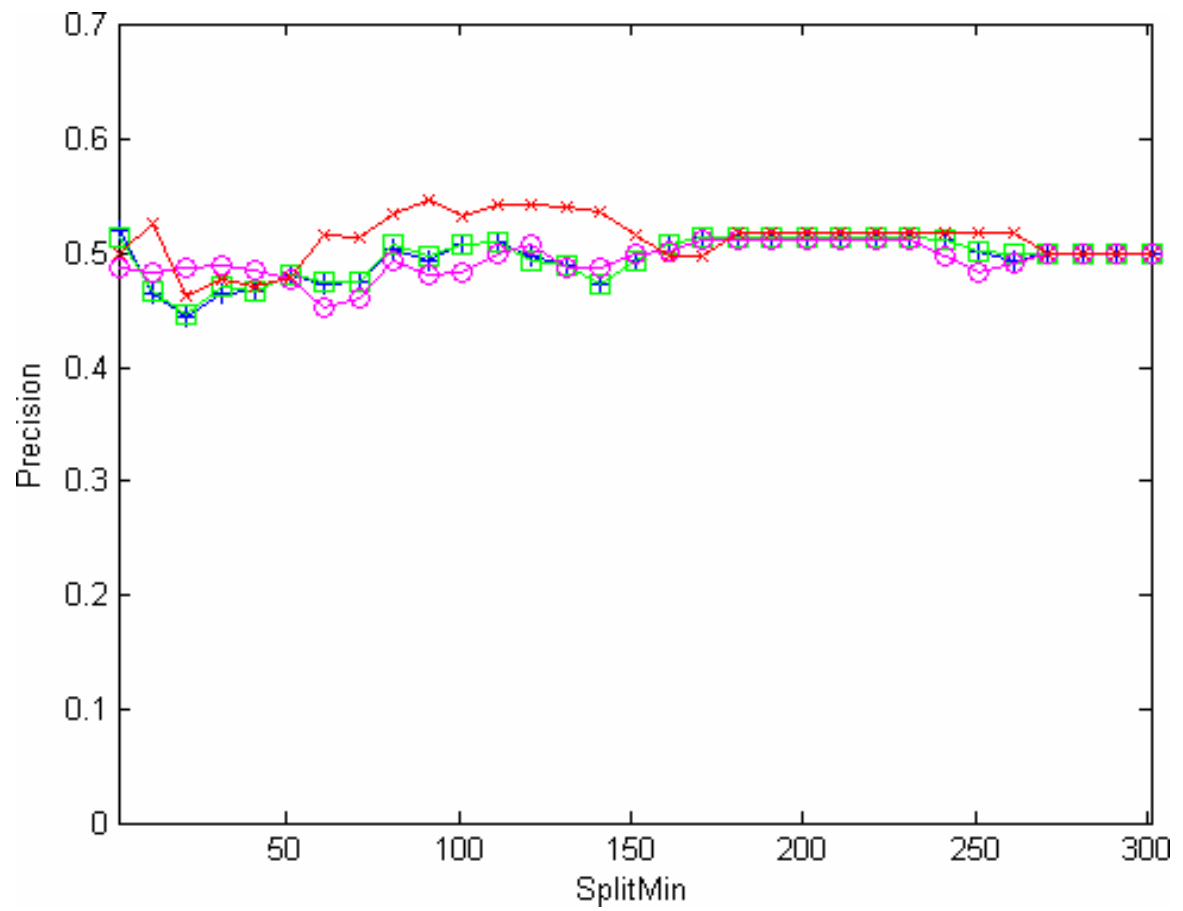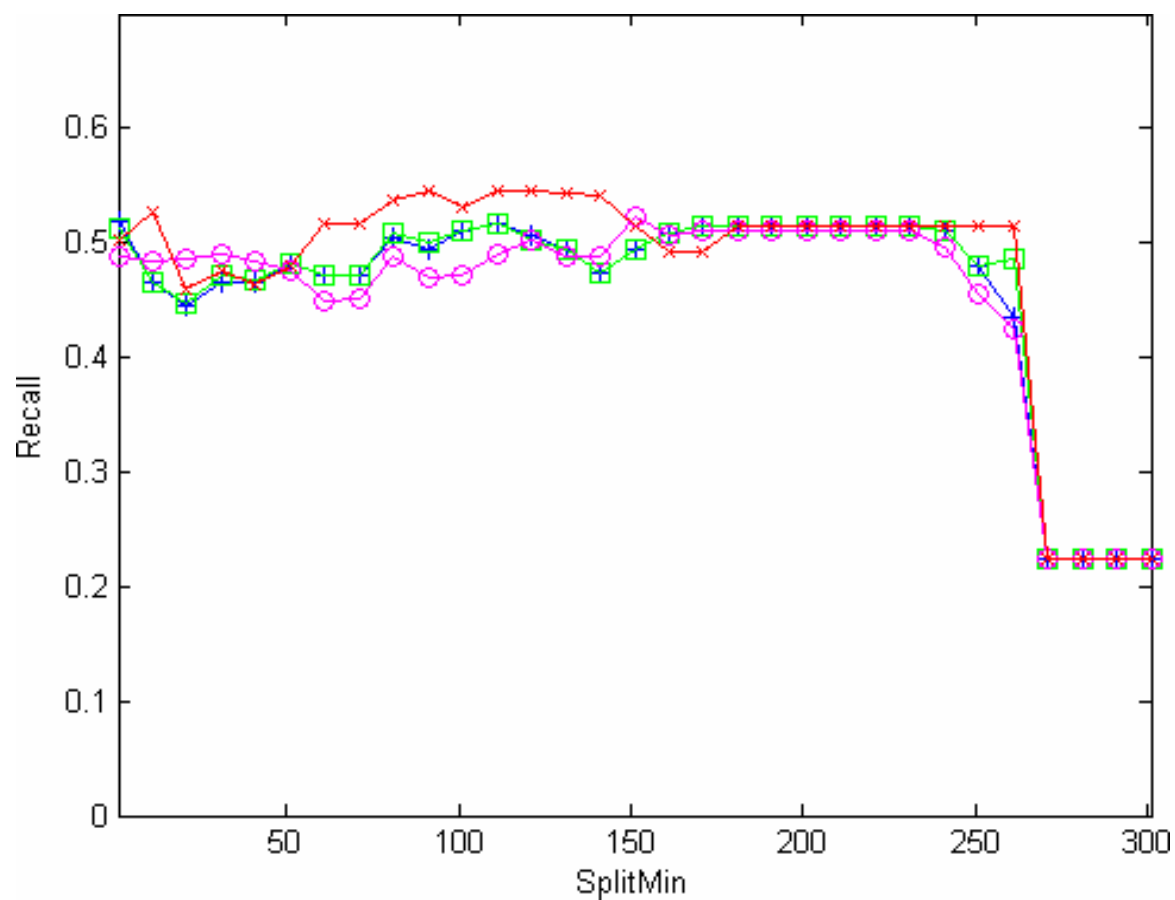

(b)

Figure 4.9 – Continued
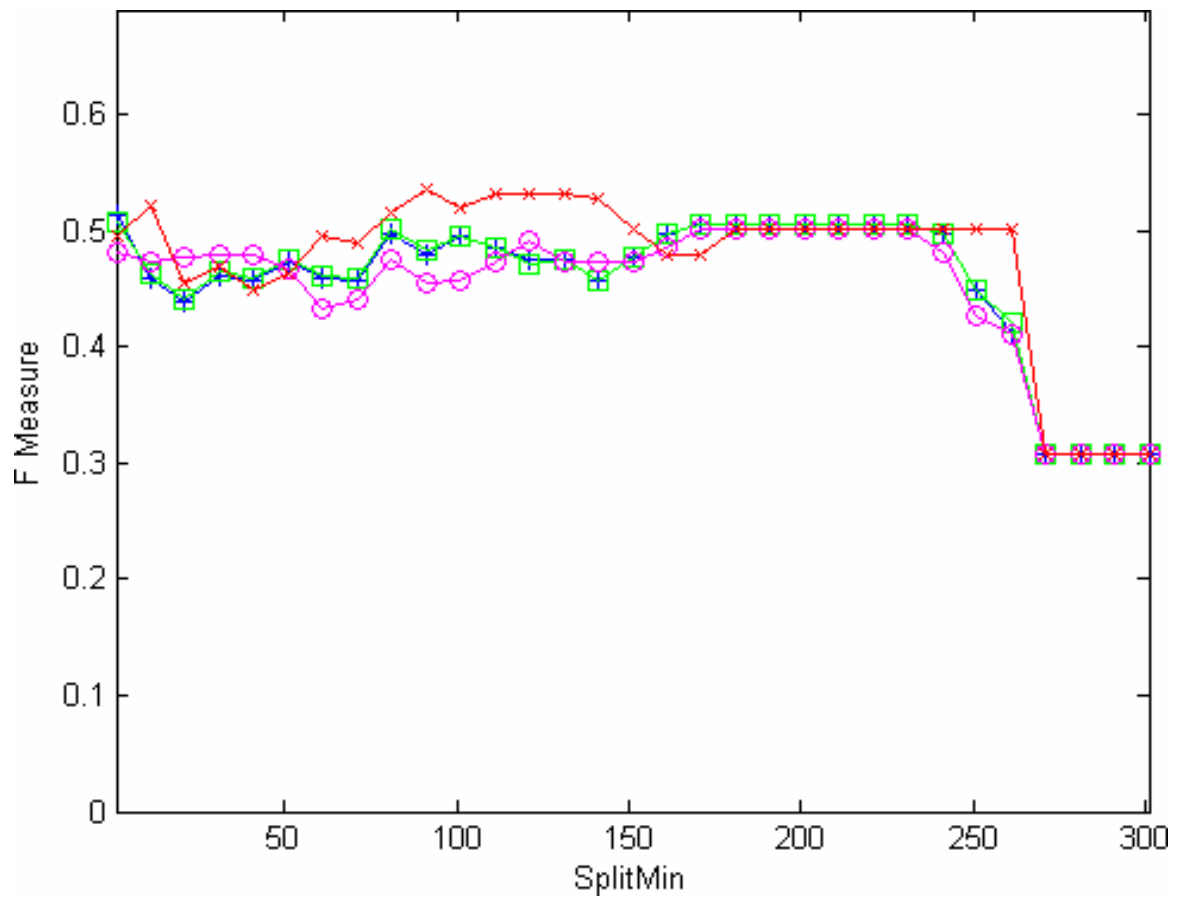

(c)

Figure 4.9 – Continued

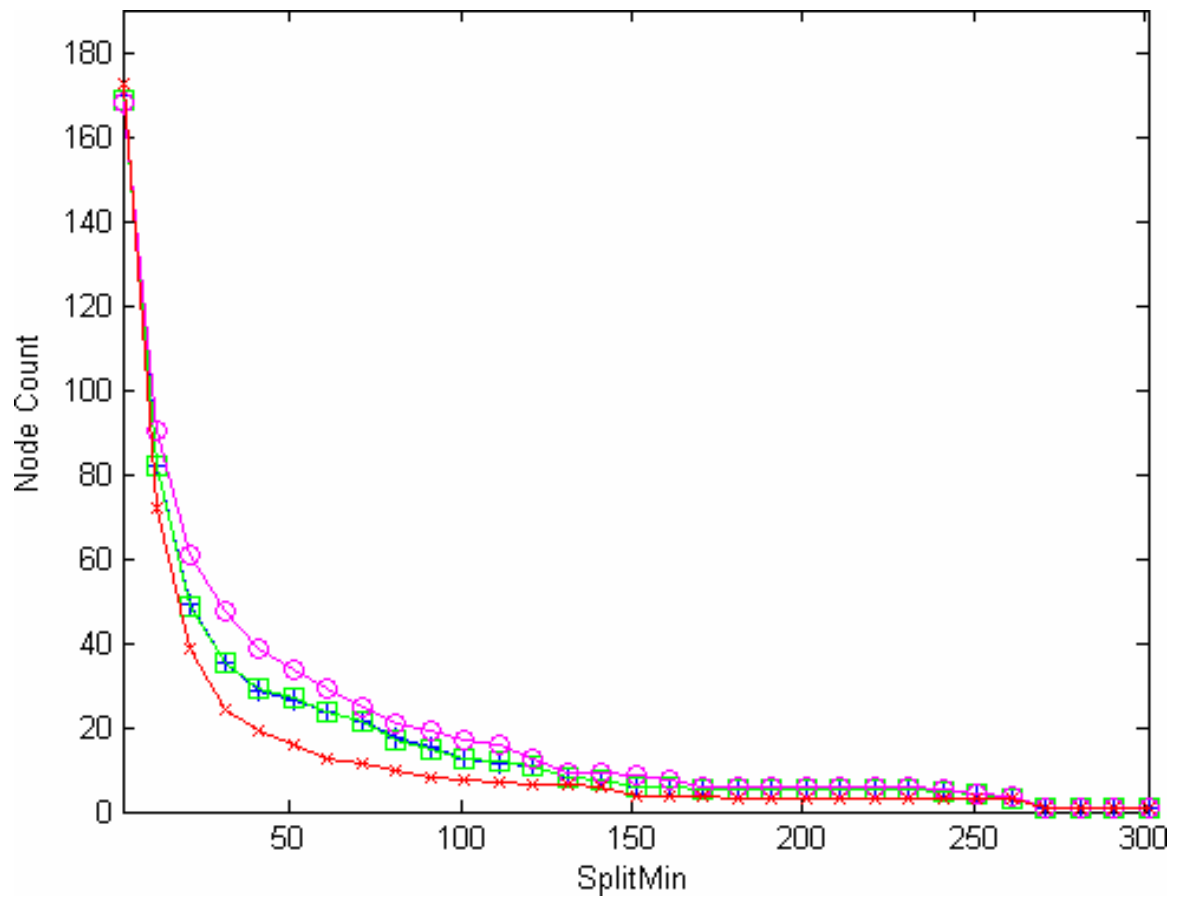(d)

Figure 4.9 – Continued

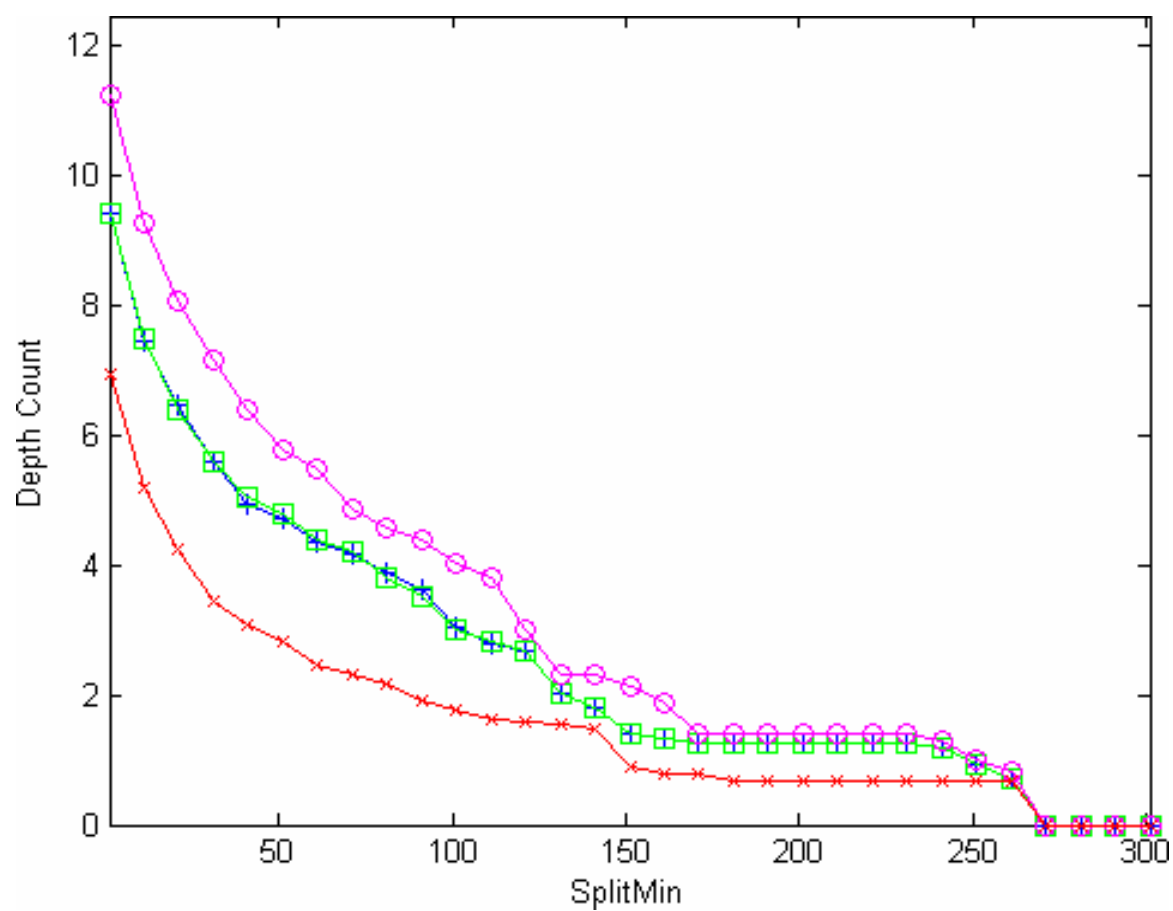(e)

Figure 4.9 – Continued

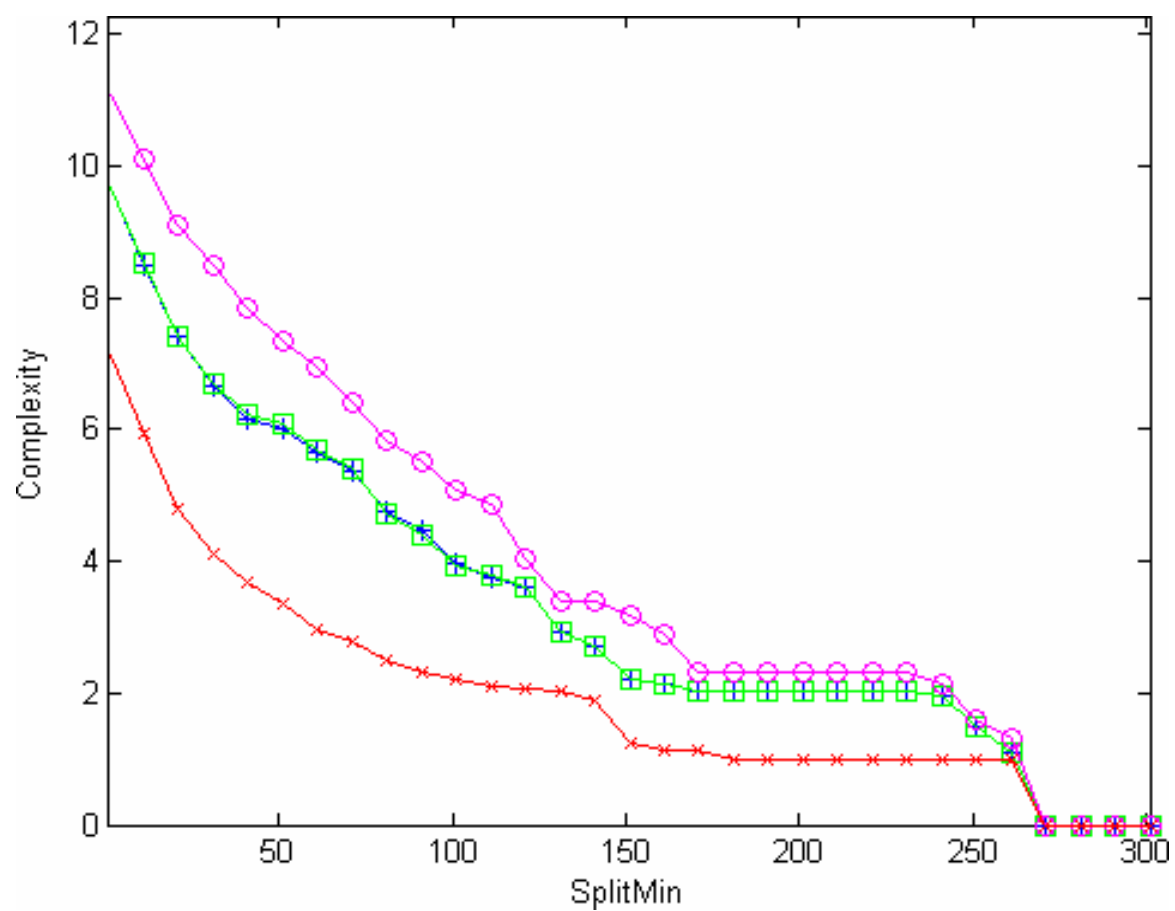
(f)

Figure 4.9 – Continued

(g)

Figure 4.9 – Continued

(h)

Figure 4.9 – Continued

(i)

criterion results at each splitmin is much smaller due to the increased density, which is consistent with the results in Experiment 3b.

The structural metric results for the Rough Product splitting criterion are significantly lower than those for the Gini Index, Twoing Rule, and MDR splitting criteria, as shown in Figure 4.10(g, h, i). Unexpectedly, we also see that the mean tree depth and mean complexity for the Rough Product did not change from Experiment 4a.

4.2.2 Real World Data Sets

In this subsection, we evaluate our splitting criteria using the following real-world data sets: the Wisconsin Breast Cancer Database, the Top Five Cancers Data Set, and the U.S. Income Data Set. Each data set contains unique characteristics that put different stresses on the splitting criteria. The Wisconsin Breast Cancer Database forces each splitting criterion to evaluate a substantial number of splits at each node before selecting the preferred split. The Top Five Cancers Data Set contains a small number of attributes that classify to several decision values. The U.S. Income Data Set provides a comprehensive experiment due to its balanced mixture of numerical and categorical attributes.

4.2.2.1 Wisconsin Breast Cancer Database

In this experiment, we use the Wisconsin Breast Cancer Database that Dr. William H. Wolberg (University of Wisconsin Hospitals) made available to the public. The data contains various breast biopsy measurements collected by oncologists through fine needle aspiration. Its objective is to identify benign or malignant breast cancers. The data set contains 699 samples, each containing ten numerical attributes (with values

111

(a)
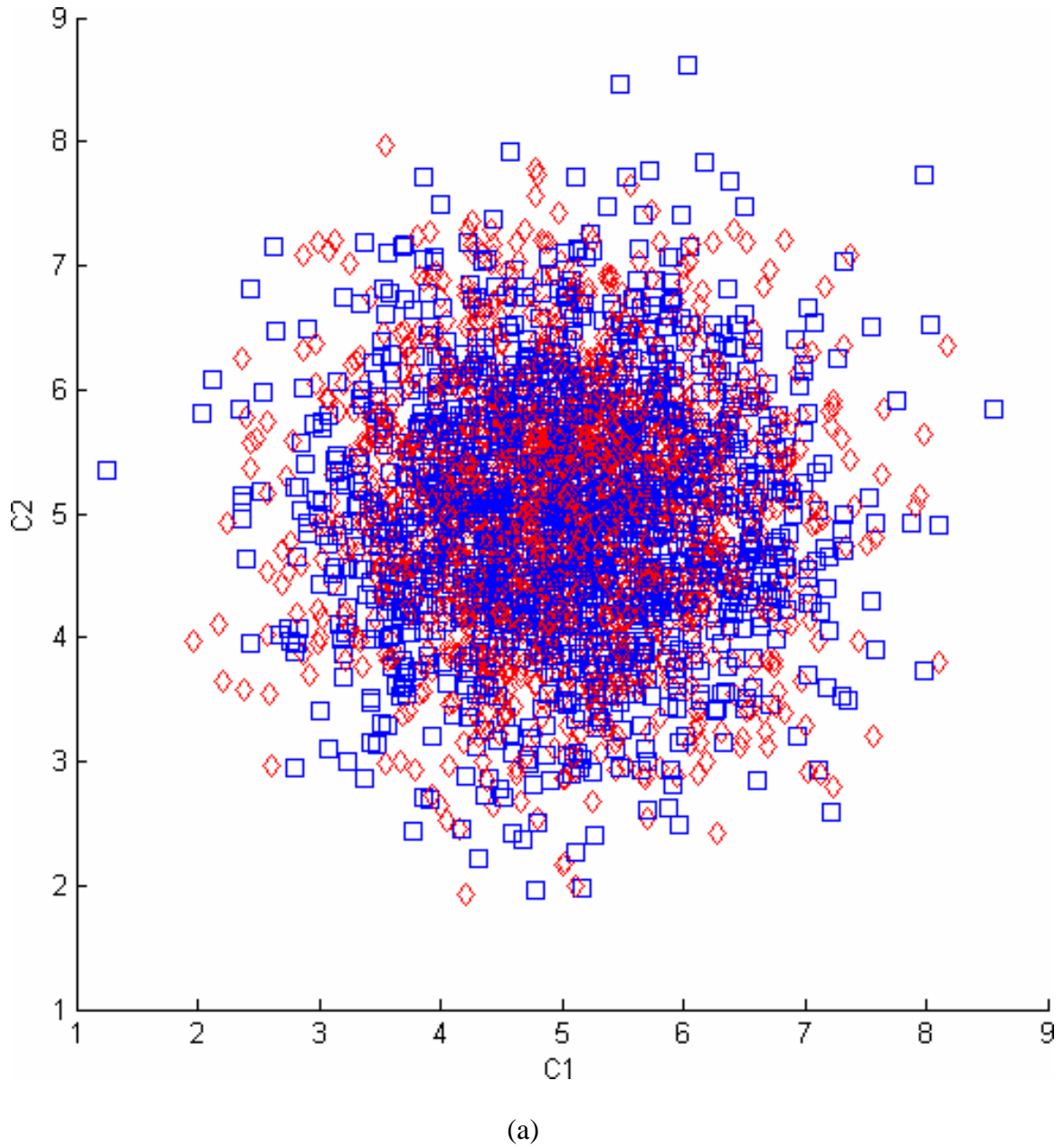
Figure 4.10: Gaussian Experiment 4b. The data set (a) is constructed such that 1500 blue three-dimensional points come from $N(5,1)$ and 1500 red three-dimensional points come from $N(5,1)$. Provided are the fold mean results for the mean accuracy (b), mean error rate (c), mean precision (d), mean recall (e), mean F-measure (f), mean node count (g), mean tree depth (h), and mean complexity (i) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)
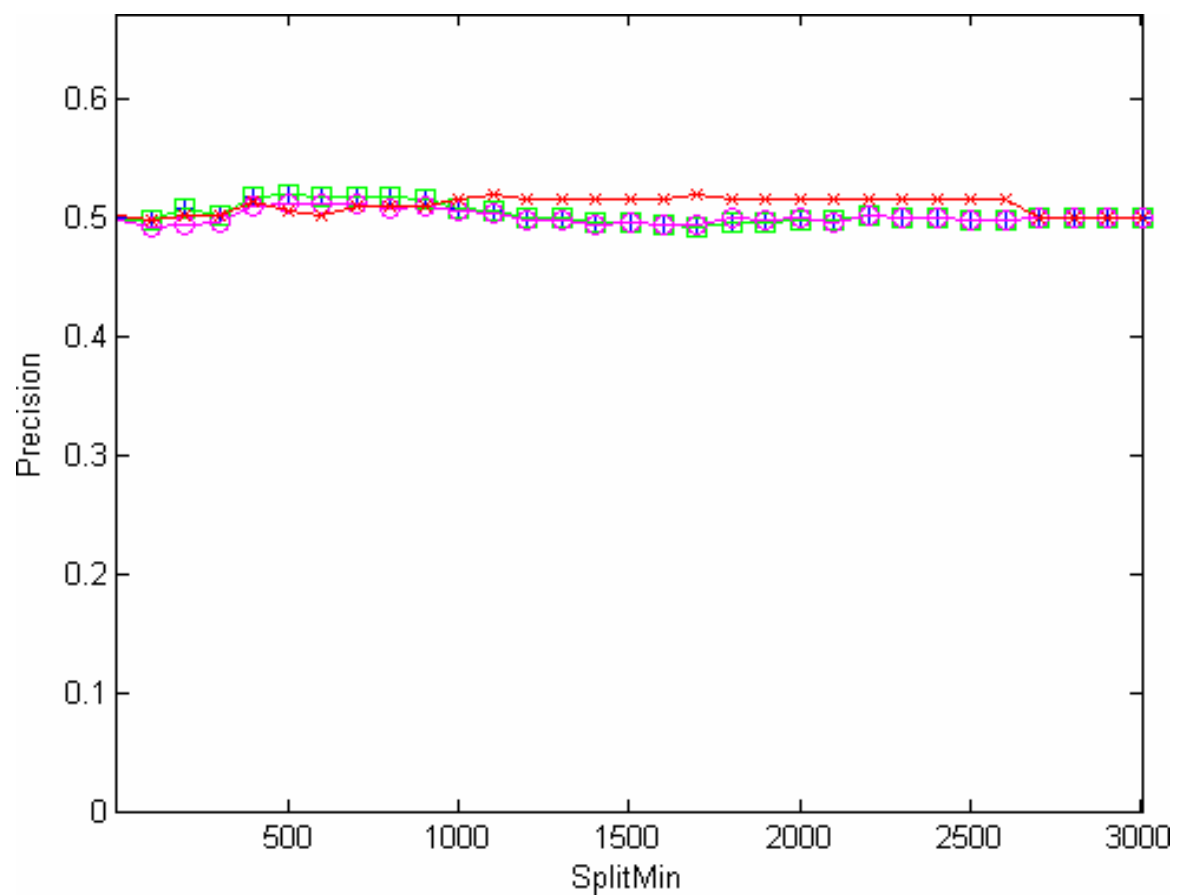
112

Figure 4.10 – Continued
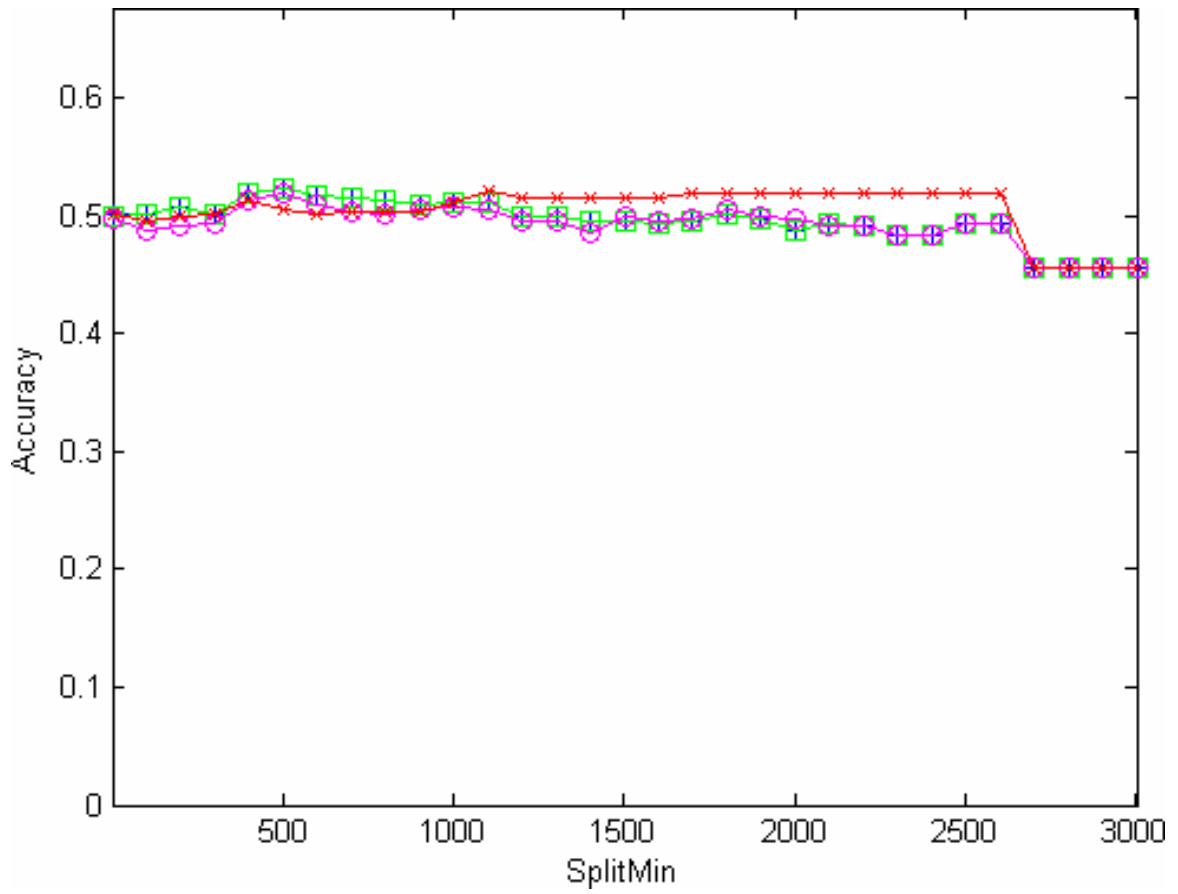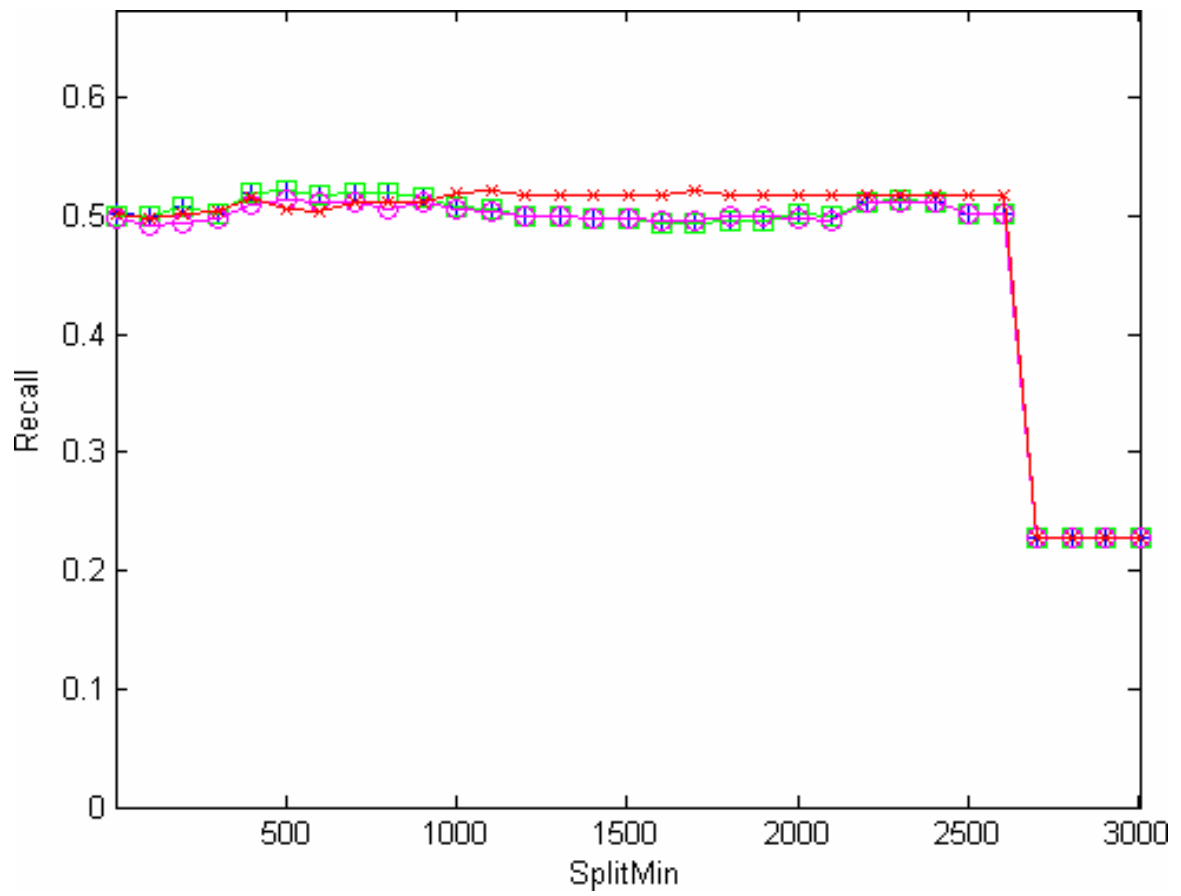

(b)

Figure 4.10 – Continued
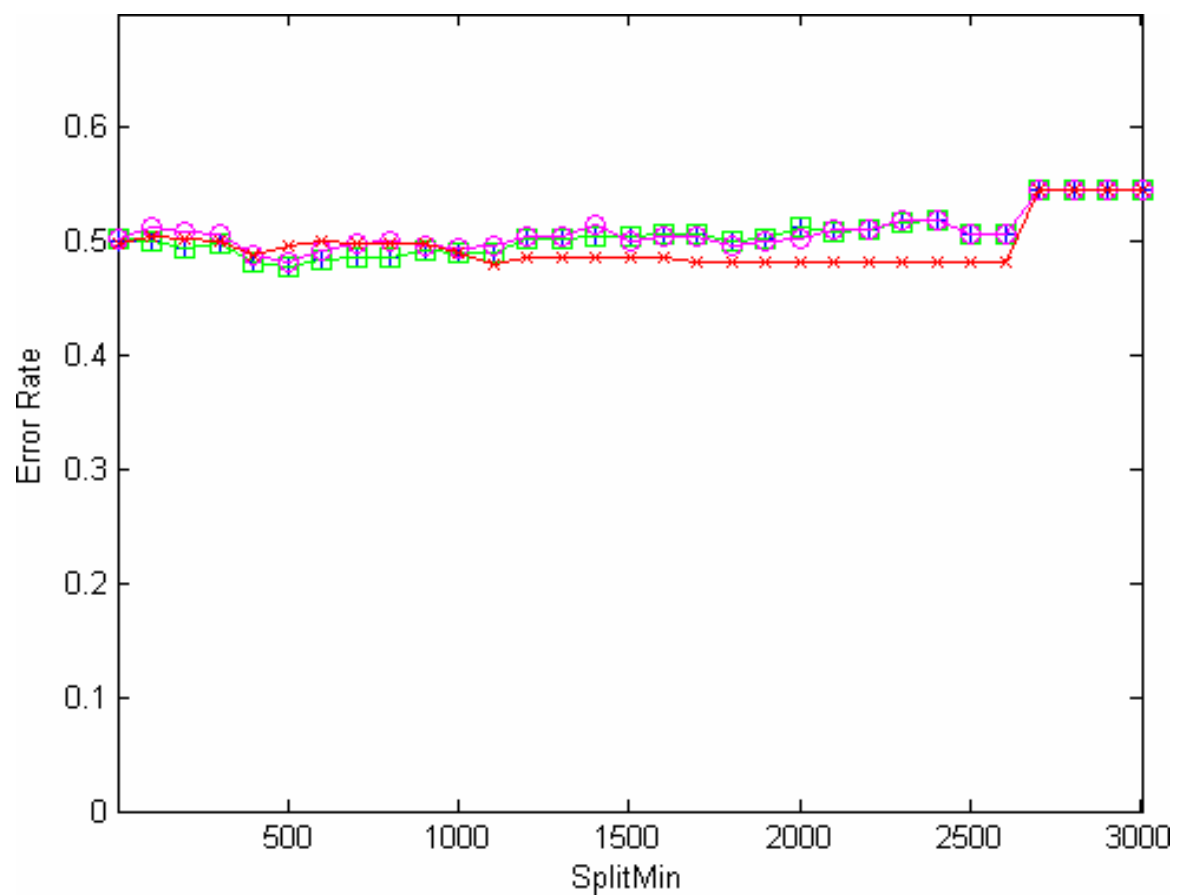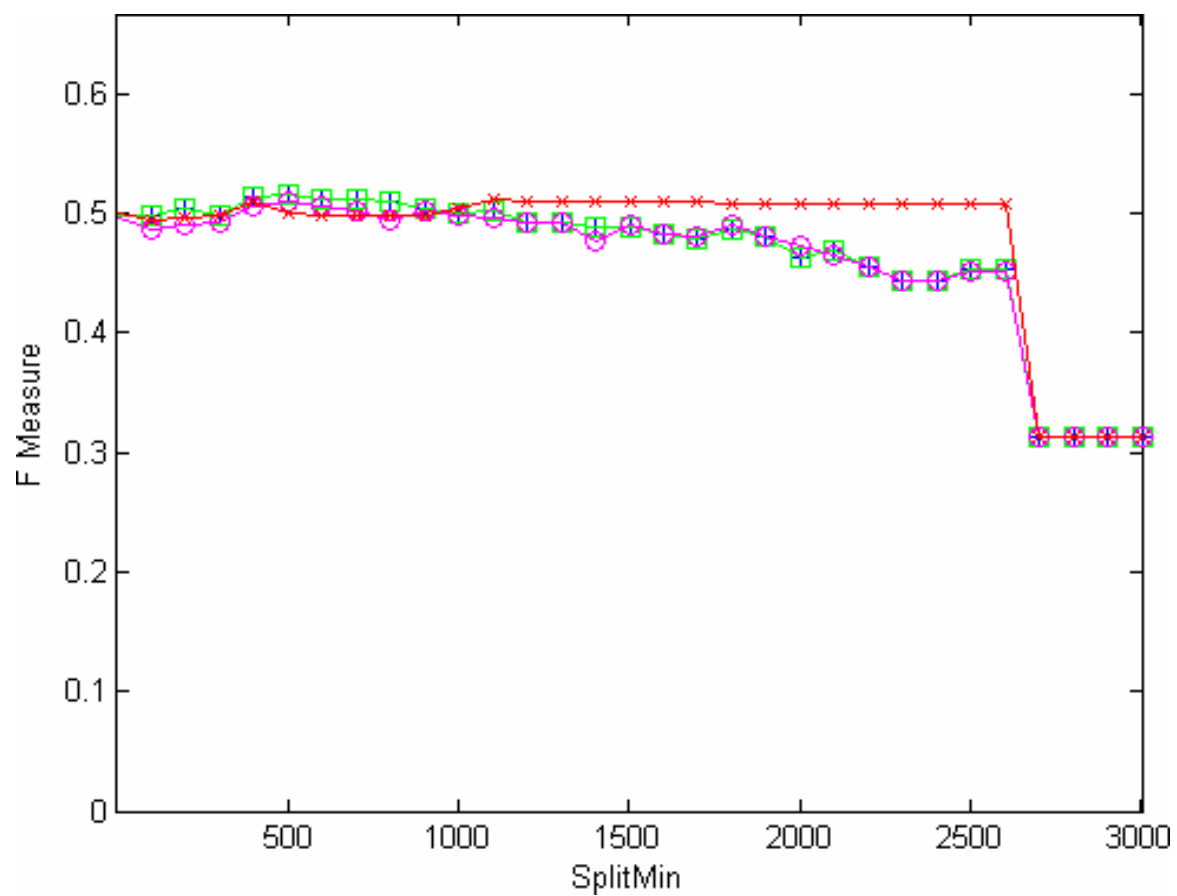

(c)

Figure 4.10 – Continued
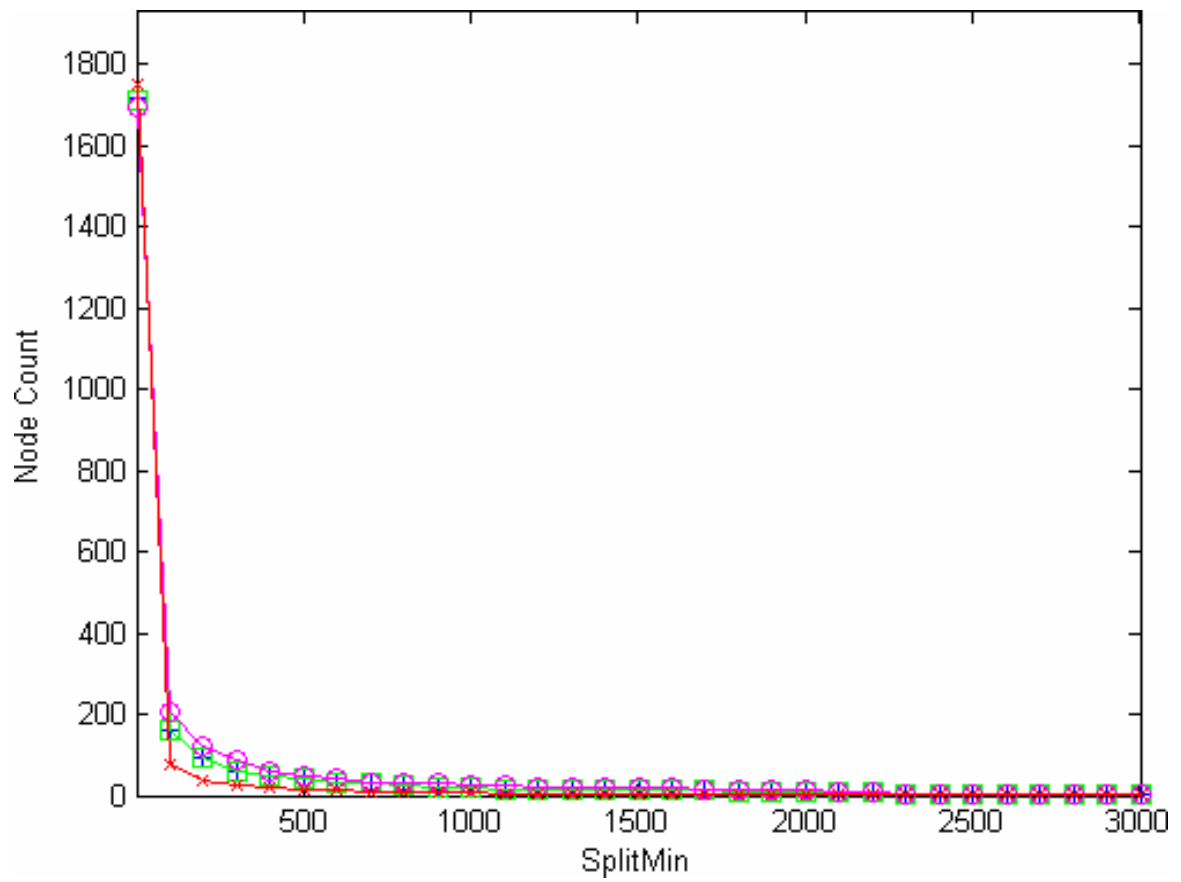

(d)

Figure 4.10 – Continued

(e)

116

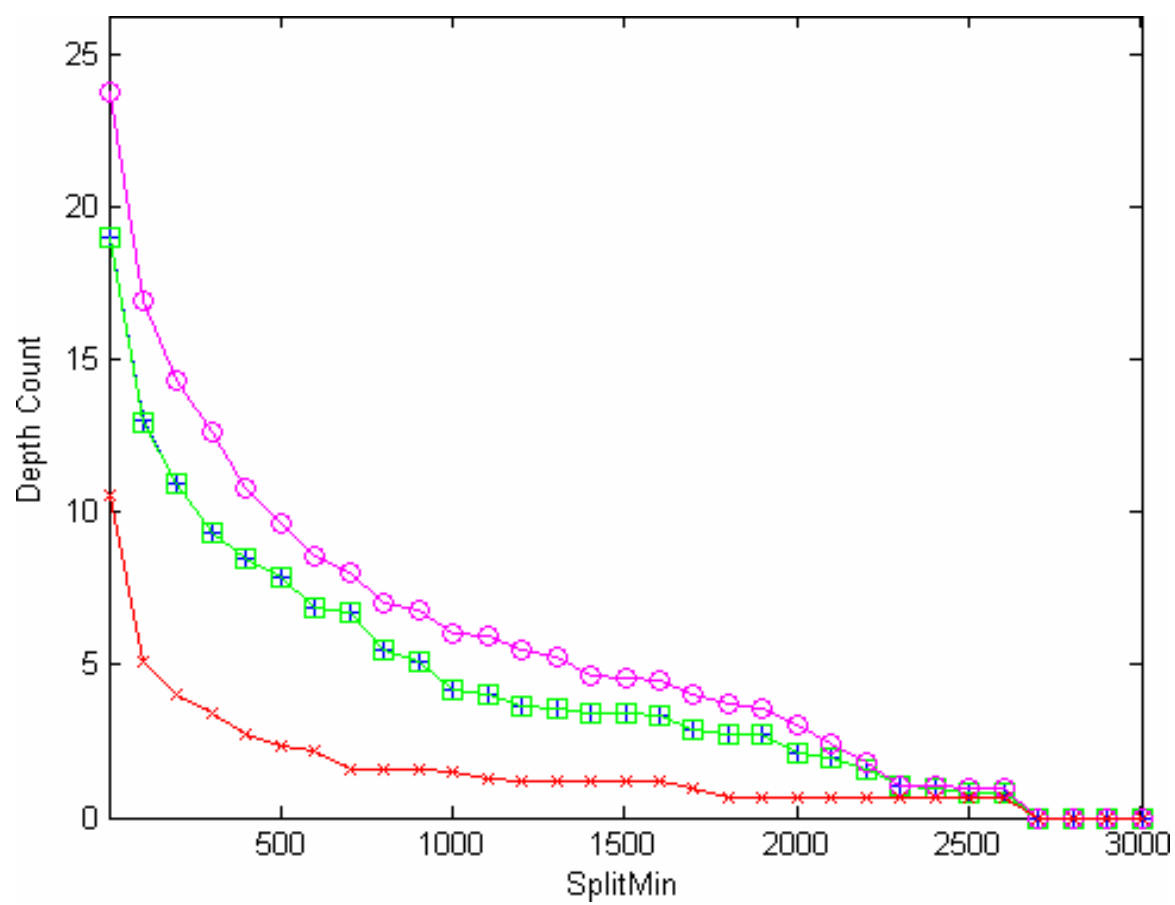Figure 4.10 – Continued

(f)

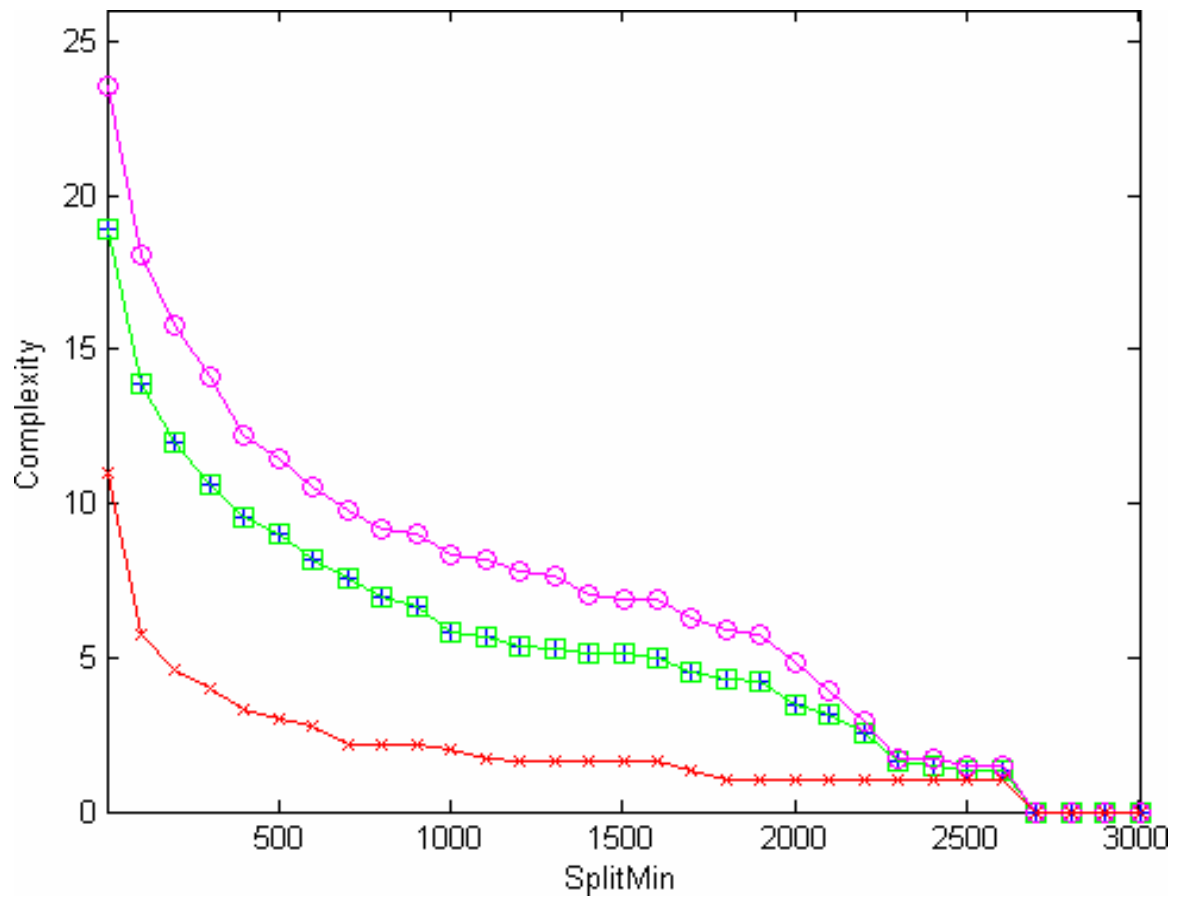Figure 4.10 – Continued

(g)

118

Figure 4.10 – Continued



(h)

Figure 4.10 – Continued

(i)

ranging from one to ten) and one categorical attribute (with values of either "benign" or "malignant"). Sixteen samples contain missing data, and therefore are not included in the experiment.

Our experiment uses 10-folds cross validation to evaluate the splitting criteria. Our splitmins range from 1 to 685 at a step interval of twenty, and provide the same training and testing sets without biases or cost functions to each tree-growing algorithm. The classification metric results, shown as Figure 4.11(a, b, c, d, e), show that the Rough Product splitting criterion produces the exact same results as the Gini Index and Twoing Rule, which, on average, are slightly better than those for the MDR splitting criterion. Structurally, however, the MDR splitting criterion generally produces decision trees with lower mean node count, mean tree depth, and mean complexity than the Gini Index, Twoing Rule, and Rough Product splitting criteria, as shown in Figure 4.11(g, h, i). As such, for this data set, we do not prefer a splitting criterion over another, as the choice is dependent on application-specific criteria. That is, the splitting criterion choice depends on whether the classification accuracy or classification speed is more important for the given application.

4.2.2.2 Top Five Cancers Data Set

In this experiment, we use Invasive Cancer Incidence data from the United States Center for Disease Control and Prevention (CDC) National Program of Cancer Registries (NPCR). The CDC provides the Wide-ranging ONline Data for Epidemiologic Research (WONDER) tool to extract publicly available data for research and statistical purposes. We extracted data that identifies one of the five most likely
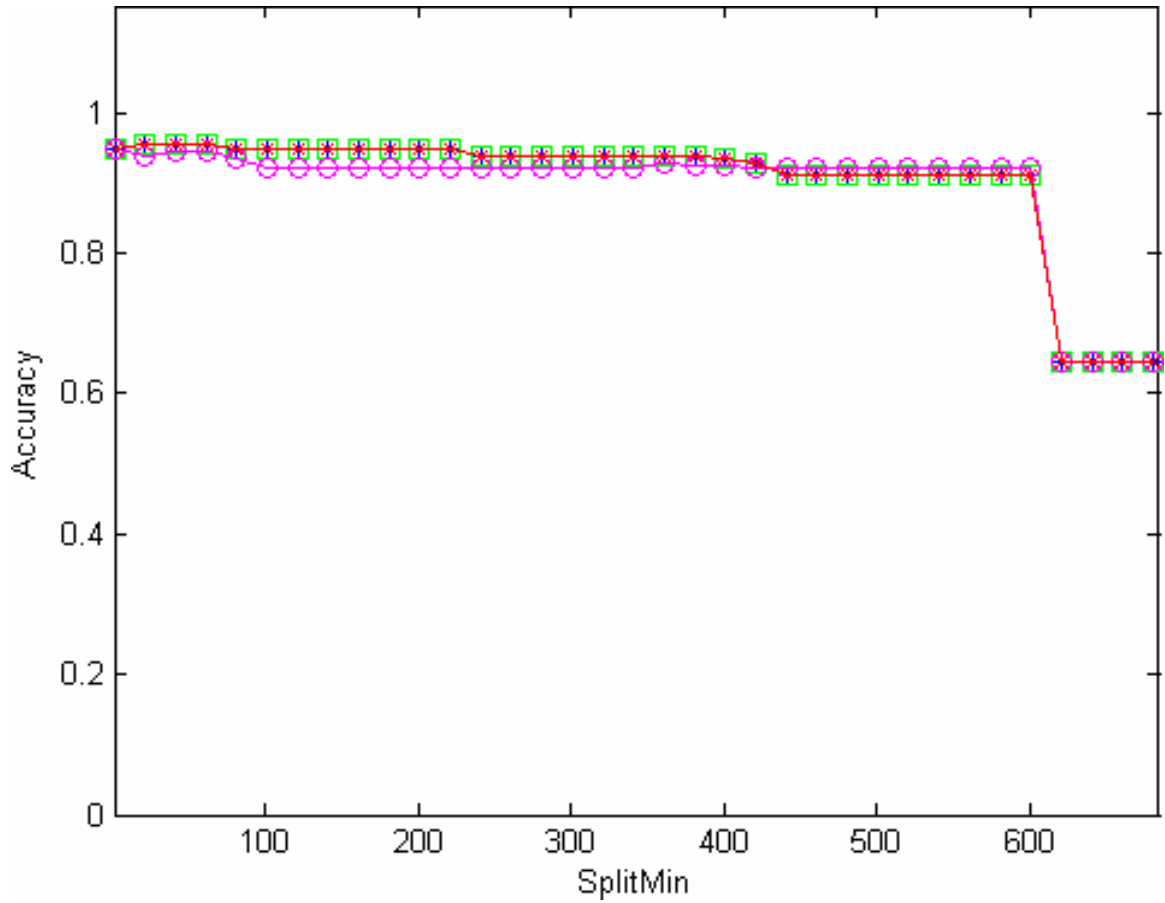
(a)

Figure 4.11: Wisconsin Breast Cancer Data Set. Provided are the fold mean results for the mean accuracy (a), mean error rate (b), mean precision (c), mean recall (d), mean F-measure (e), mean node count (f), mean tree depth (g), and mean complexity (h) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

122

Figure 4.11 – Continued
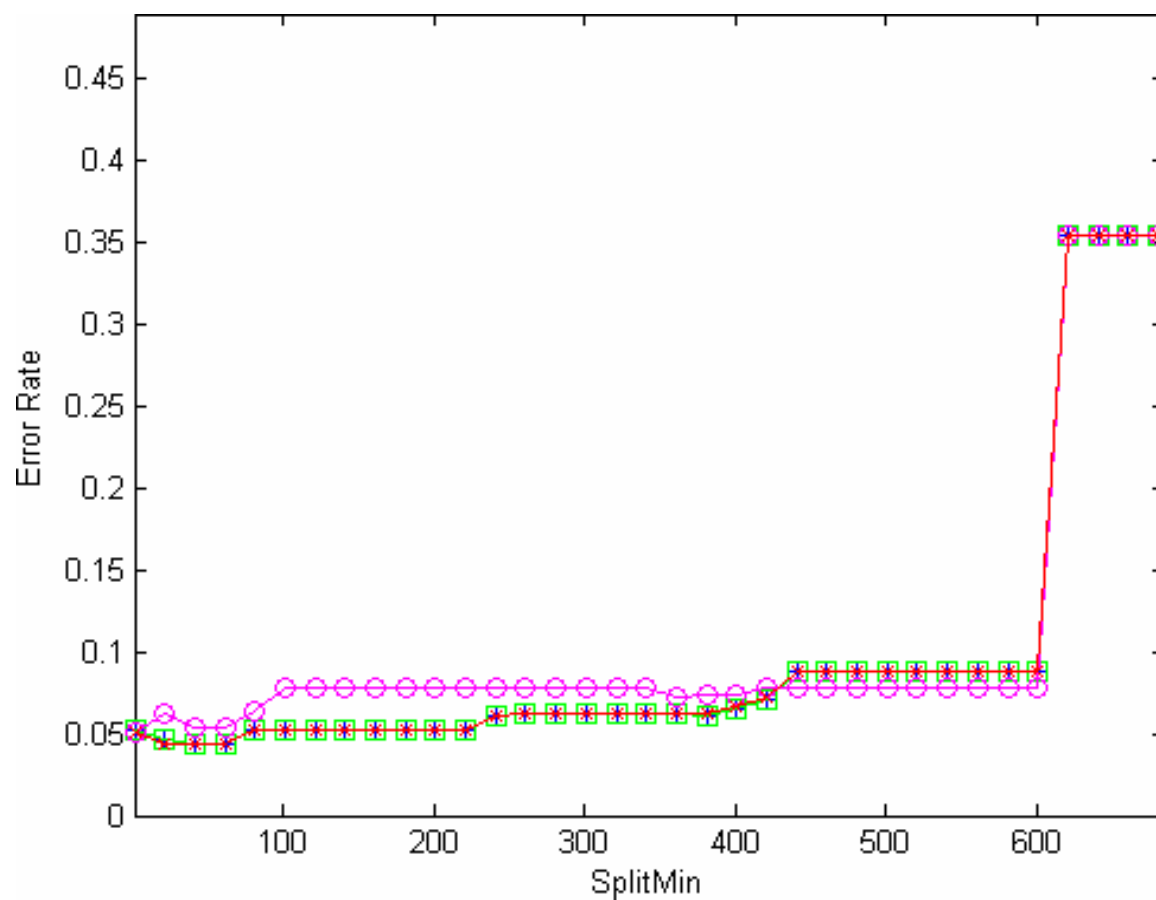
(b)

Figure 4.11 – Continued
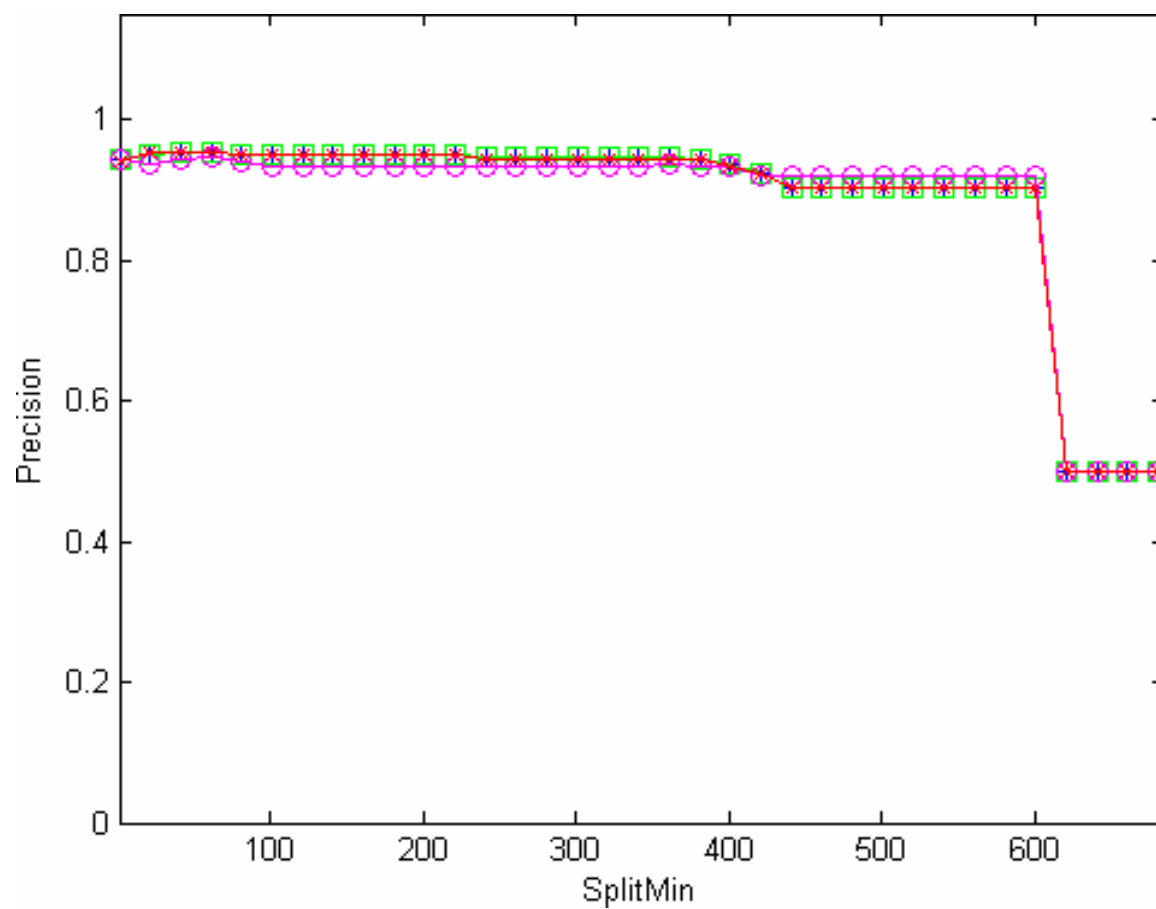
(c)

Figure 4.11 – Continued
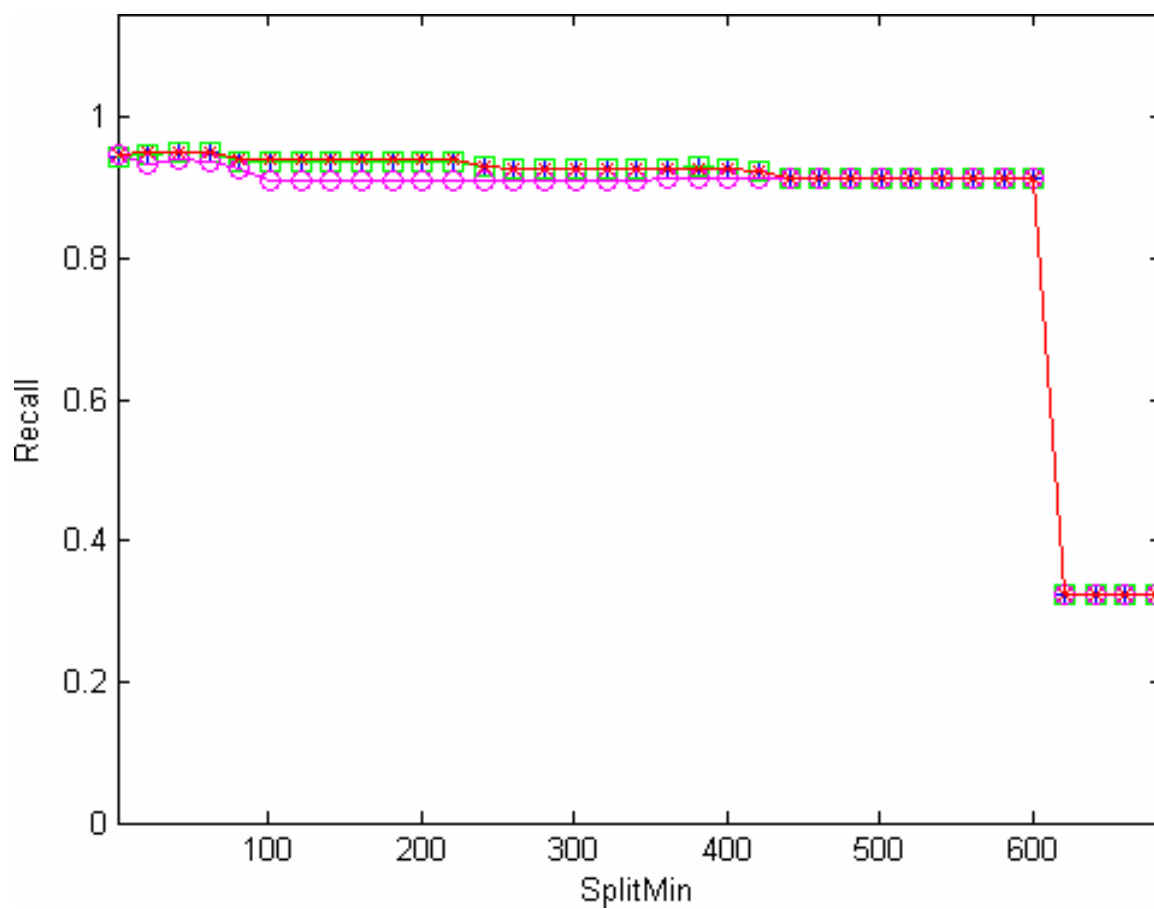
(d)

Figure 4.11 – Continued
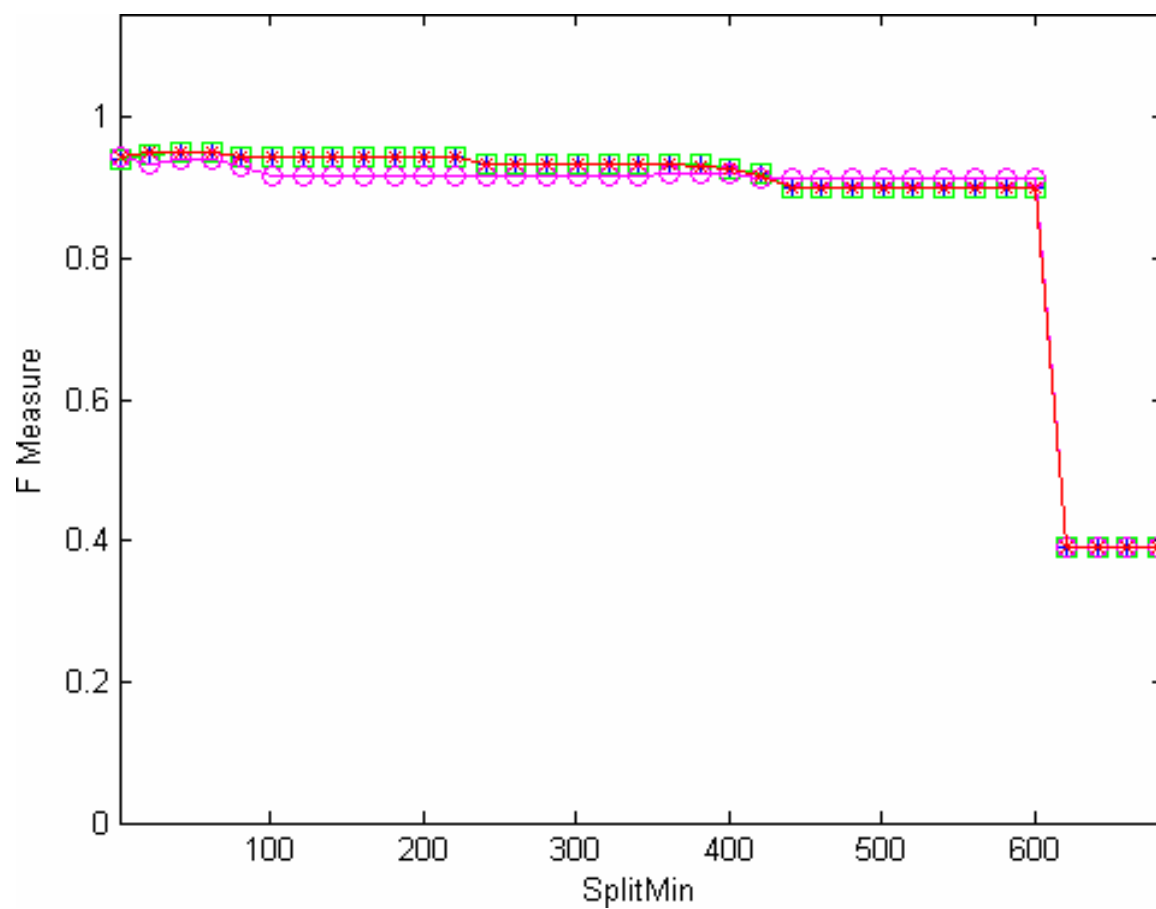
(e)

Figure 4.11 – Continued
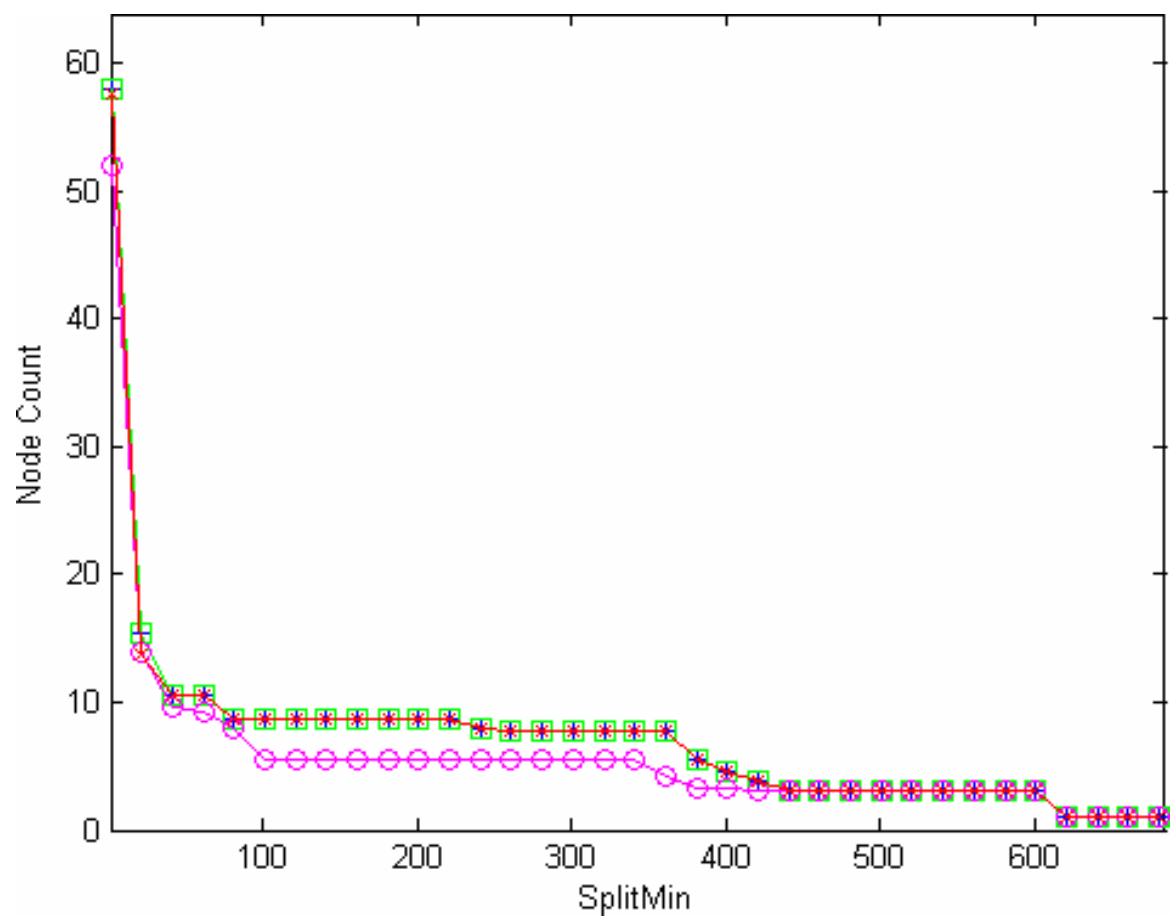
(f)

Figure 4.11 – Continued

(g)

128

Figure 4.11 – Continued
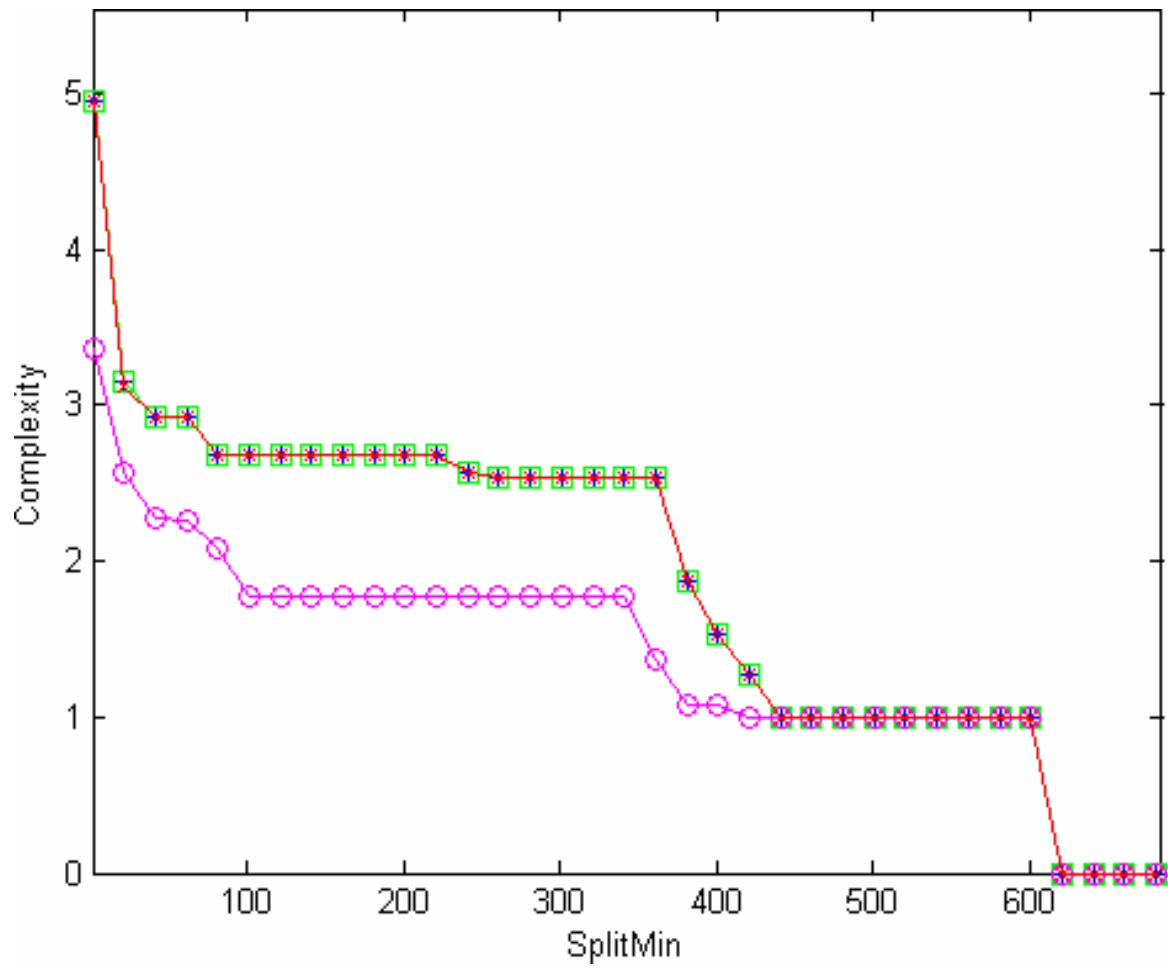
(h)

locations of cancer ("Breast," "Lung," "Ovary," "Prostate," or "Stomach") from a small

number of attributes about an individual. Our data set contains 2897 samples, each

containing one numerical attribute (with values that belong to the range of positive

whole numbers) and three categorical attributes (Race, Gender, and Age Range).

Our experiment uses 10-folds cross validation to evaluate the splitting criteria.

We cover splitmins that range from 1 to 2601 at a step interval of 100, and provide the

same training and testing sets without biases or cost functions to each tree-growing

algorithm.

The classification metric results, as shown in Figure 4.12(a, b, c, d, e), show that

none of the splitting criteria produce useful decision tree classifiers. We conclude this

by observing the small amounts of variance and generally flat shape in the mean

accuracy plot (Figure 4.12a). The variance indicates that no splitting criteria perform

prominently better while the flat shape suggests that additional splits do not

significantly improve the overall tree quality. We make this conclusion despite several

interesting results for the MDR splitting criterion between splitmins 801 and 1201 in

Figure 4.12(c, d, e). In this range, the MDR's mean precision, recall, and F-measure

values are significantly greater than the corresponding values in the Gini Index, Twoing

Rule, and Rough Product splitting criteria. However, these results give false

impressions. A closer inspection of this splitmin range indicates that the MDR splitting

criterion produces a considerably high percentage of true positive counts for the

"Ovary" value, whereas the Gini Index, Twoing Rule, and Rough Product do not. This

result is perplexing because the Maximum Deviance Reduction does not exhibit a

significant boost in accuracy relative to the other criteria. However, we discover that,
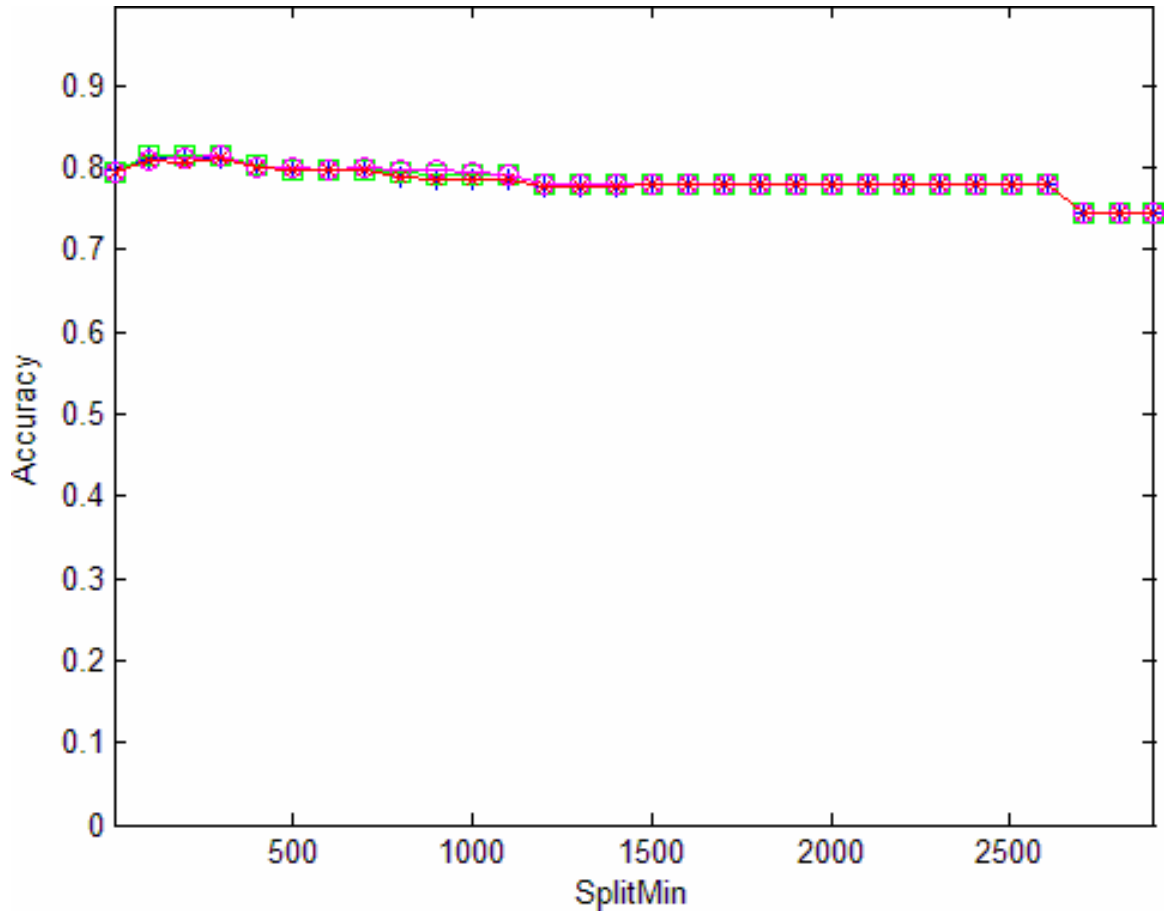
(a)

Figure 4.12: Five Cancers Data Set. Provided are the fold mean results for the mean accuracy (a), mean error rate (b), mean precision (c), mean recall (d), mean F-measure (e), mean node count (f), mean tree depth (g), and mean complexity (h) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.12 – Continued
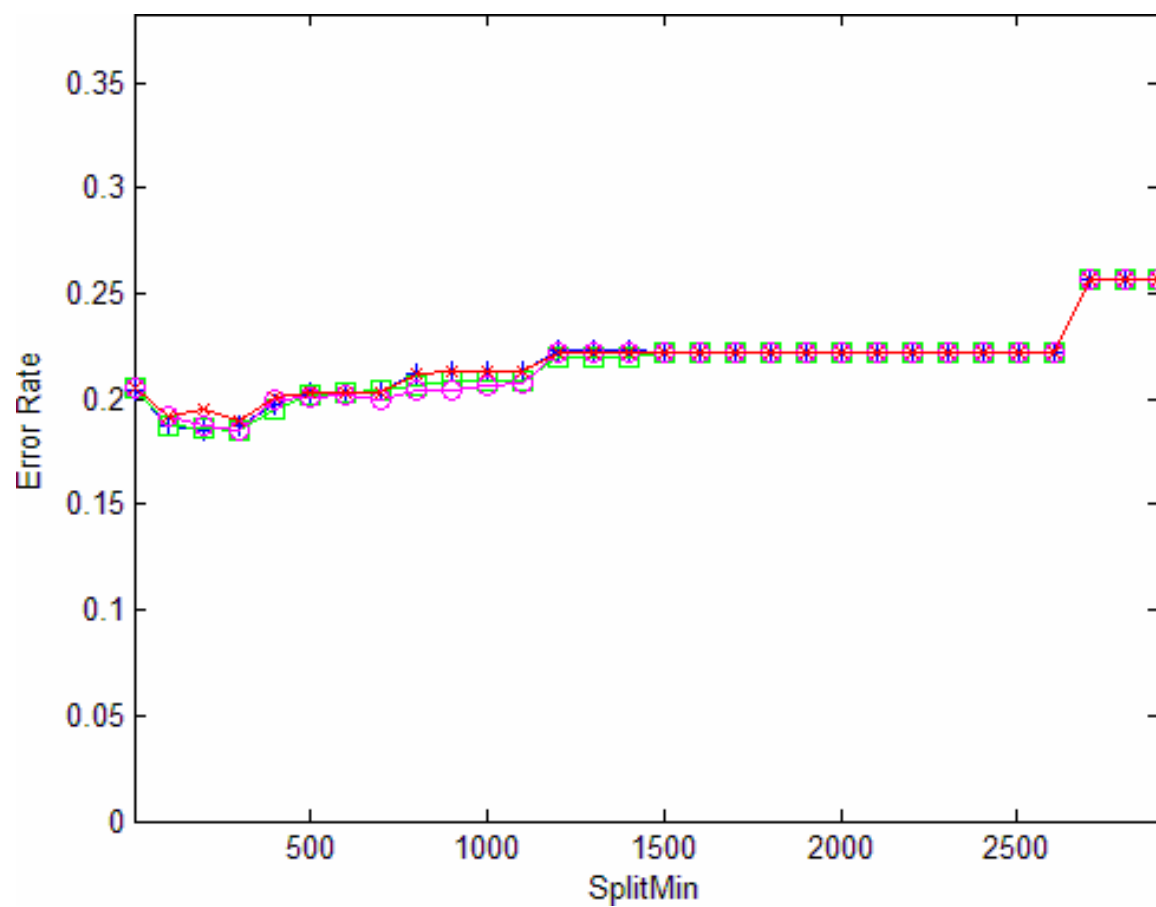
(b)

Figure 4.12 – Continued
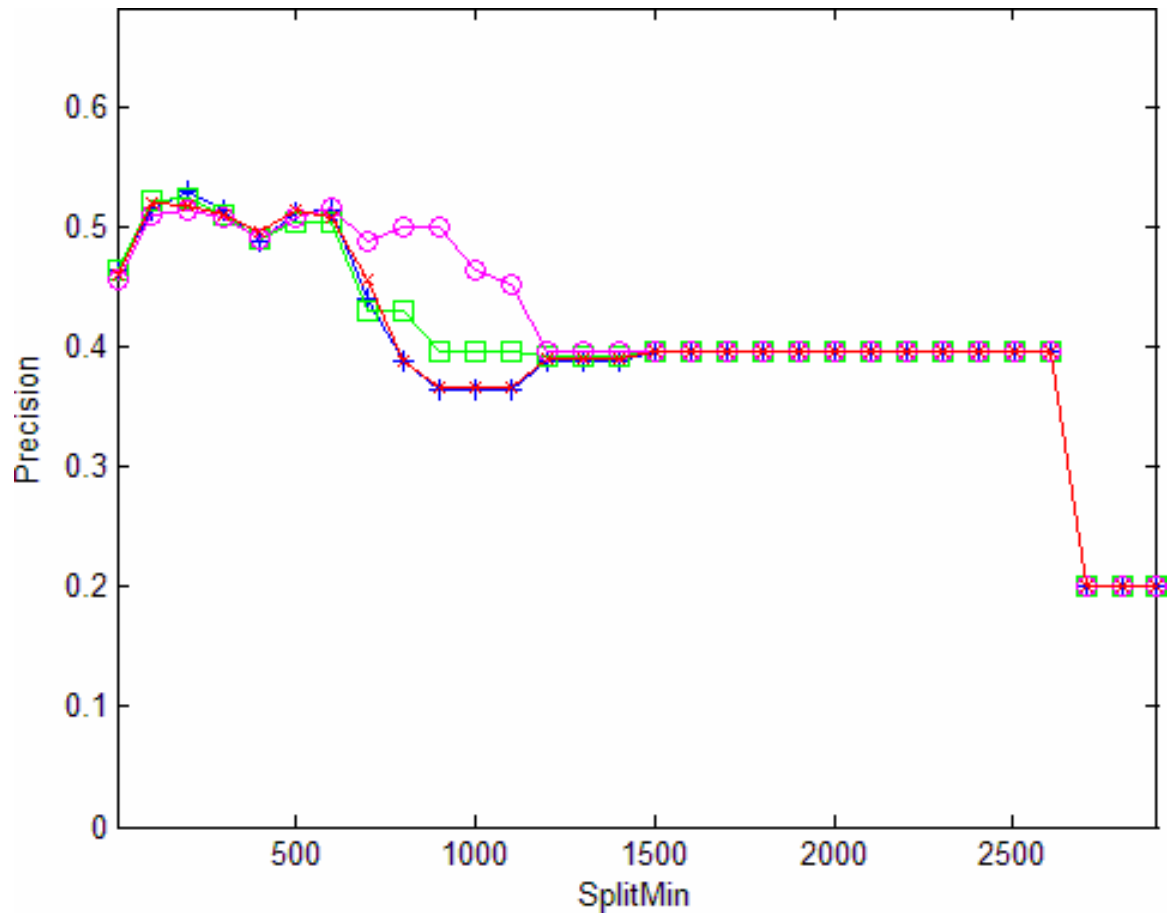
(c)

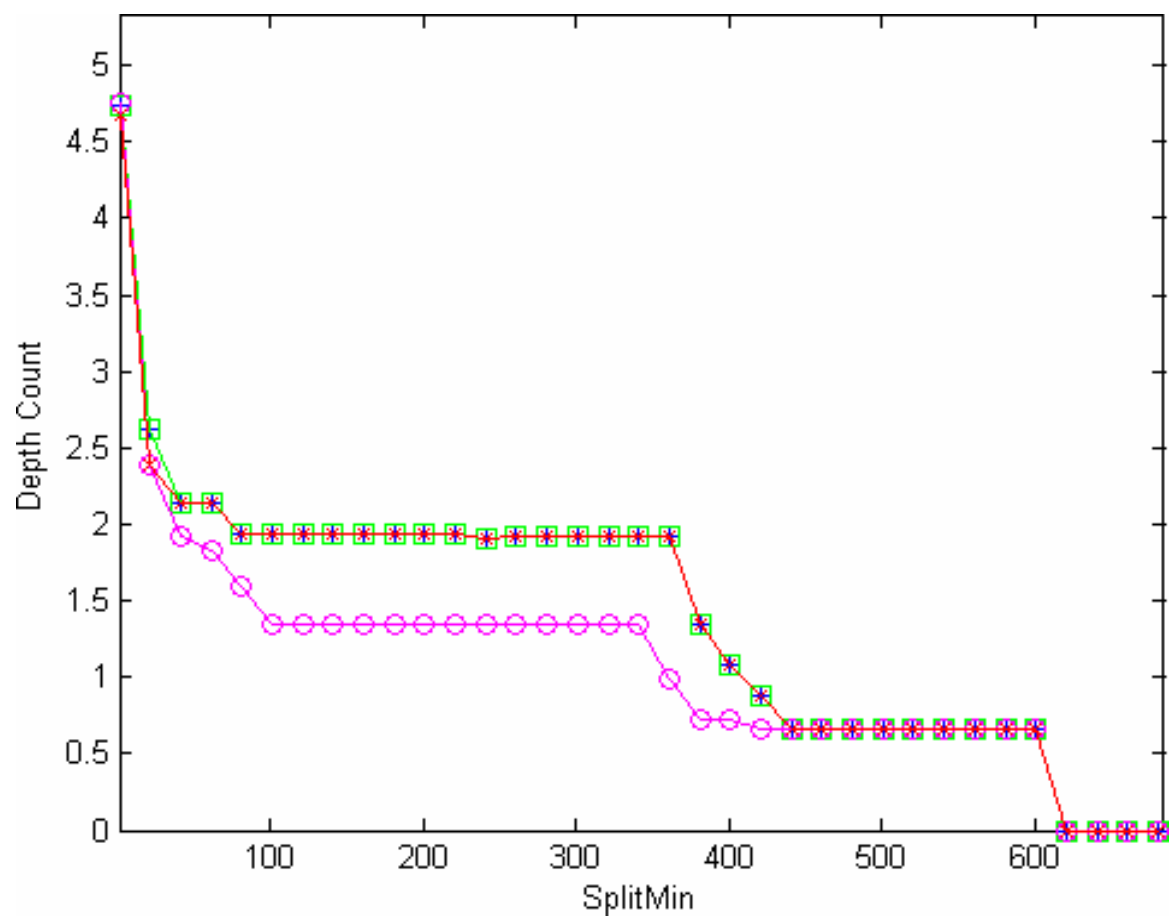Figure 4.12 – Continued
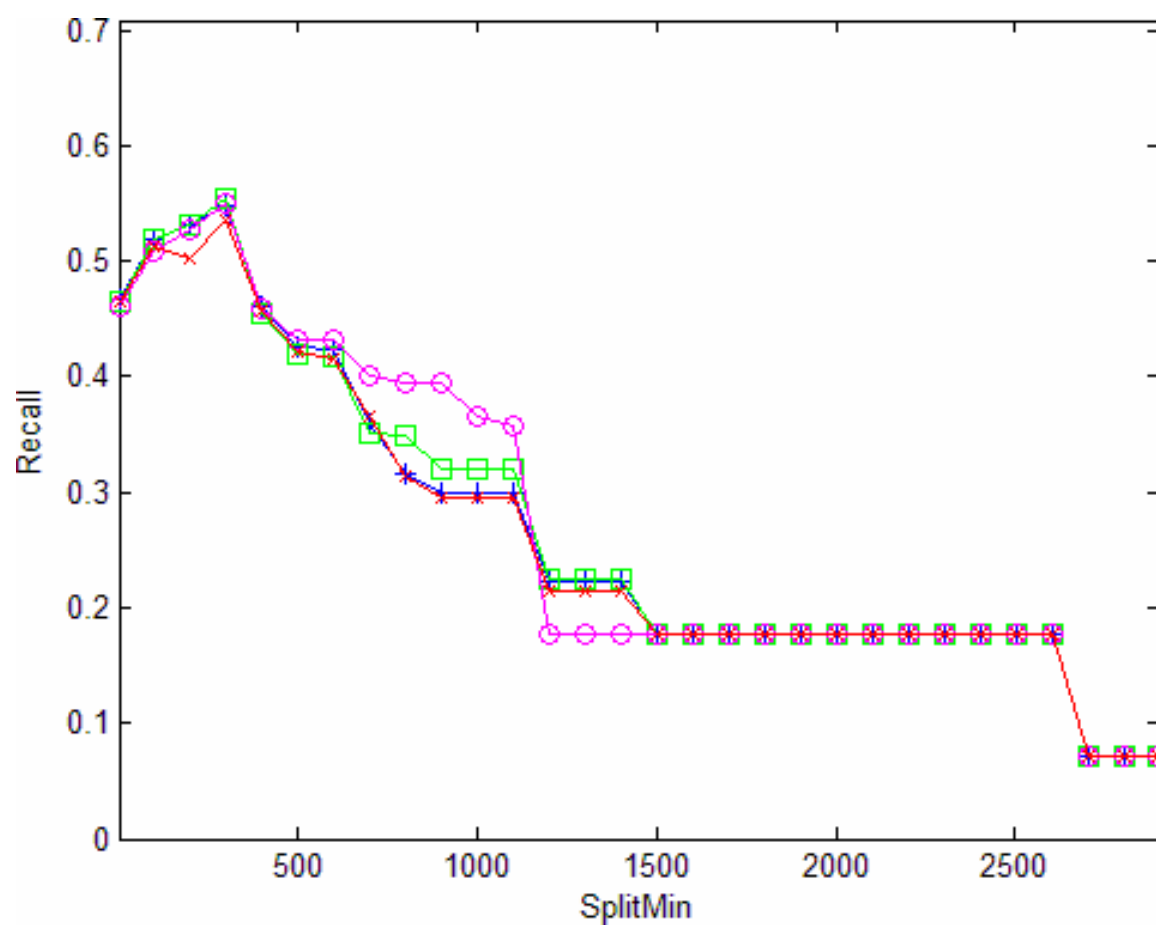
(d)

Figure 4.12 – Continued
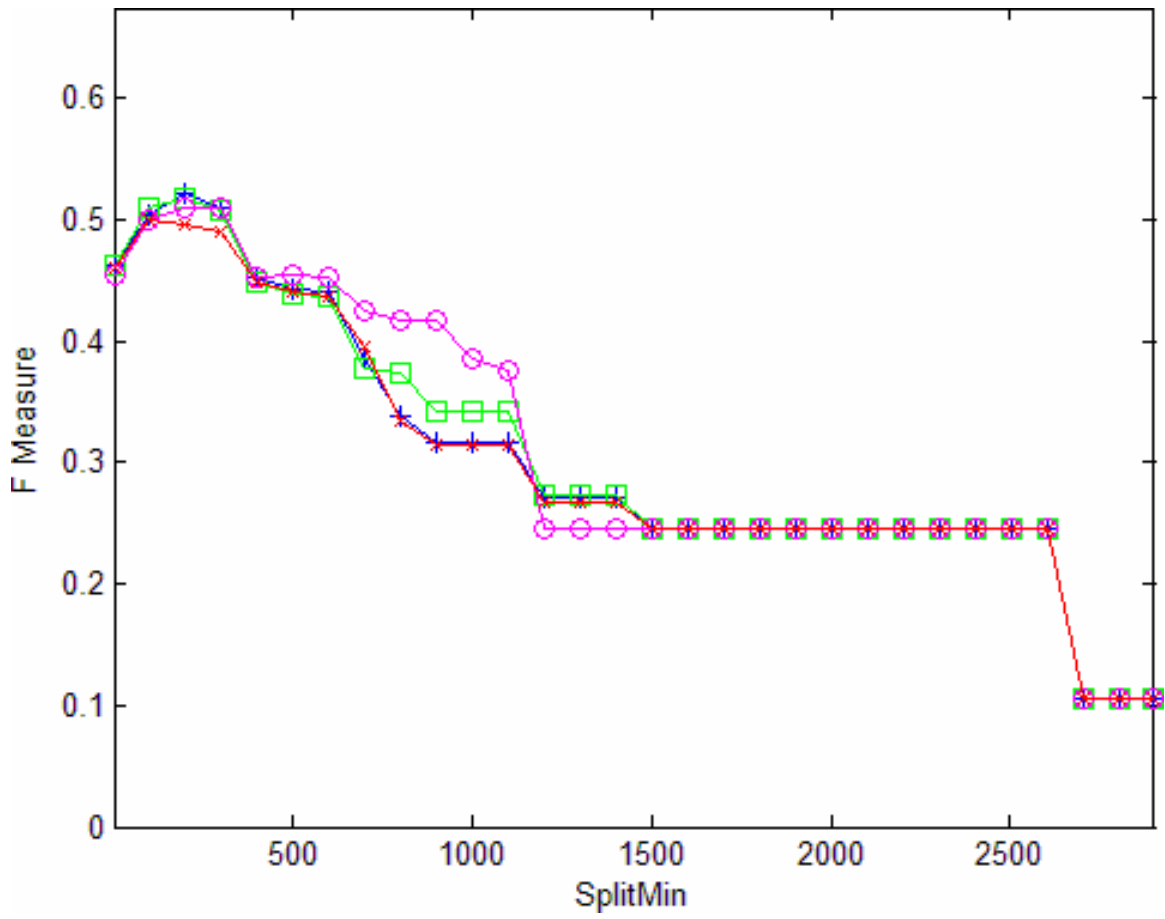
(e)

Figure 4.12 – Continued
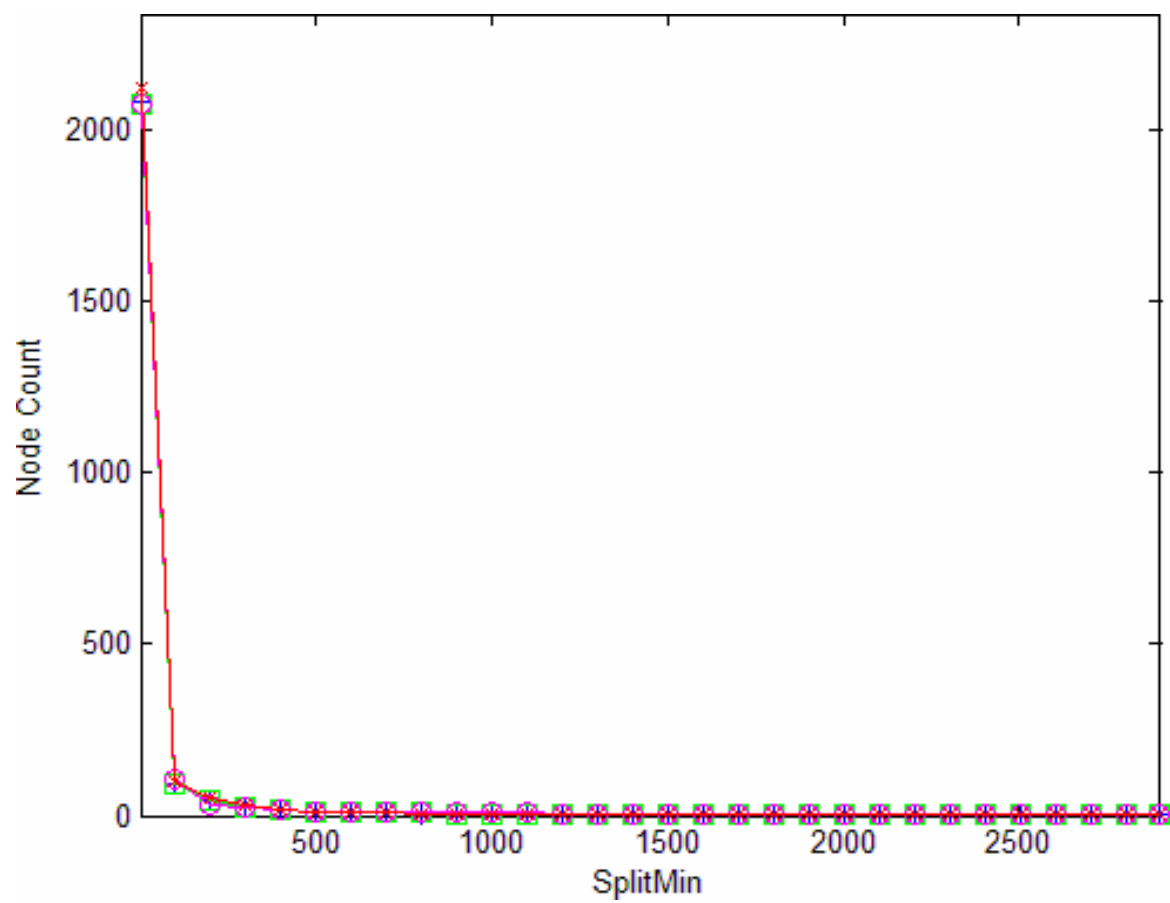
(f)

Figure 4.12 – Continued
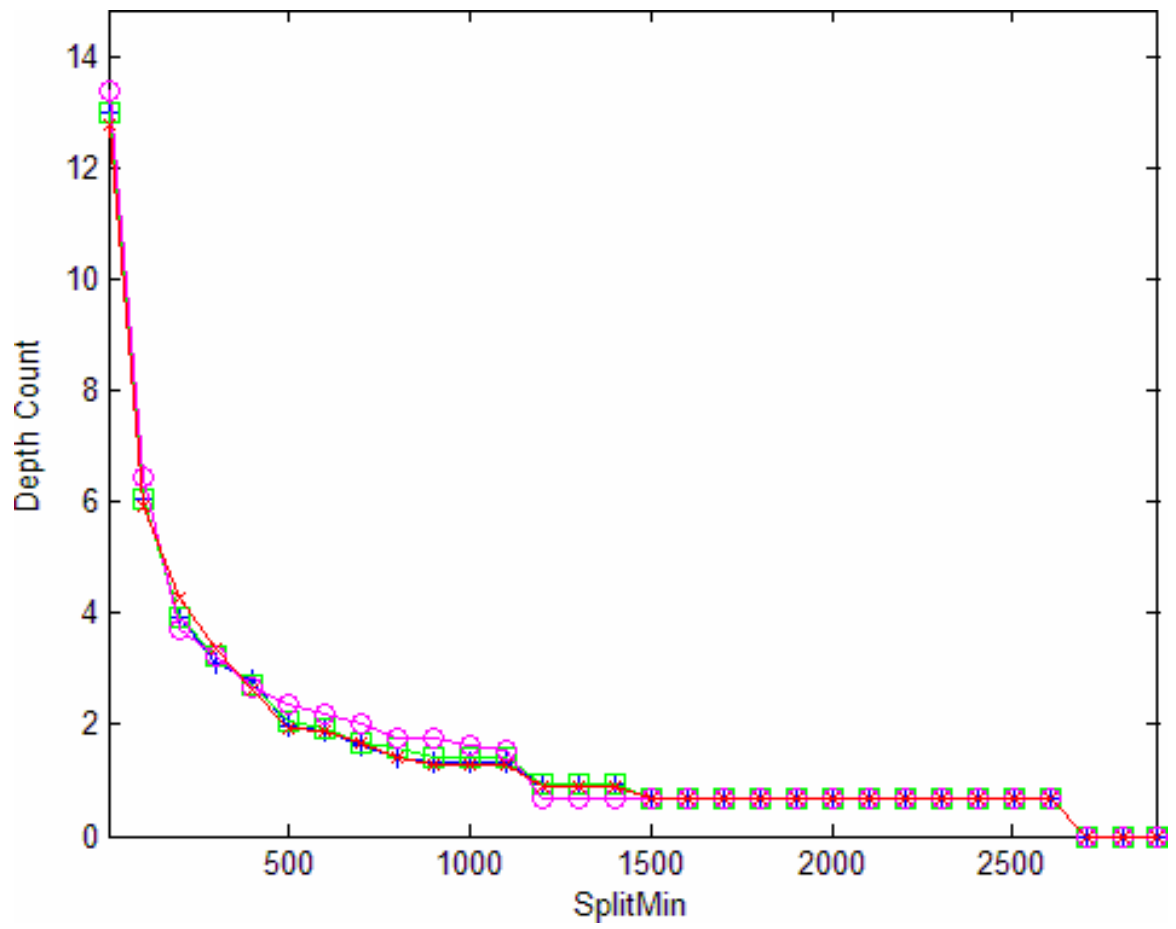
(g)

Figure 4.12 – Continued
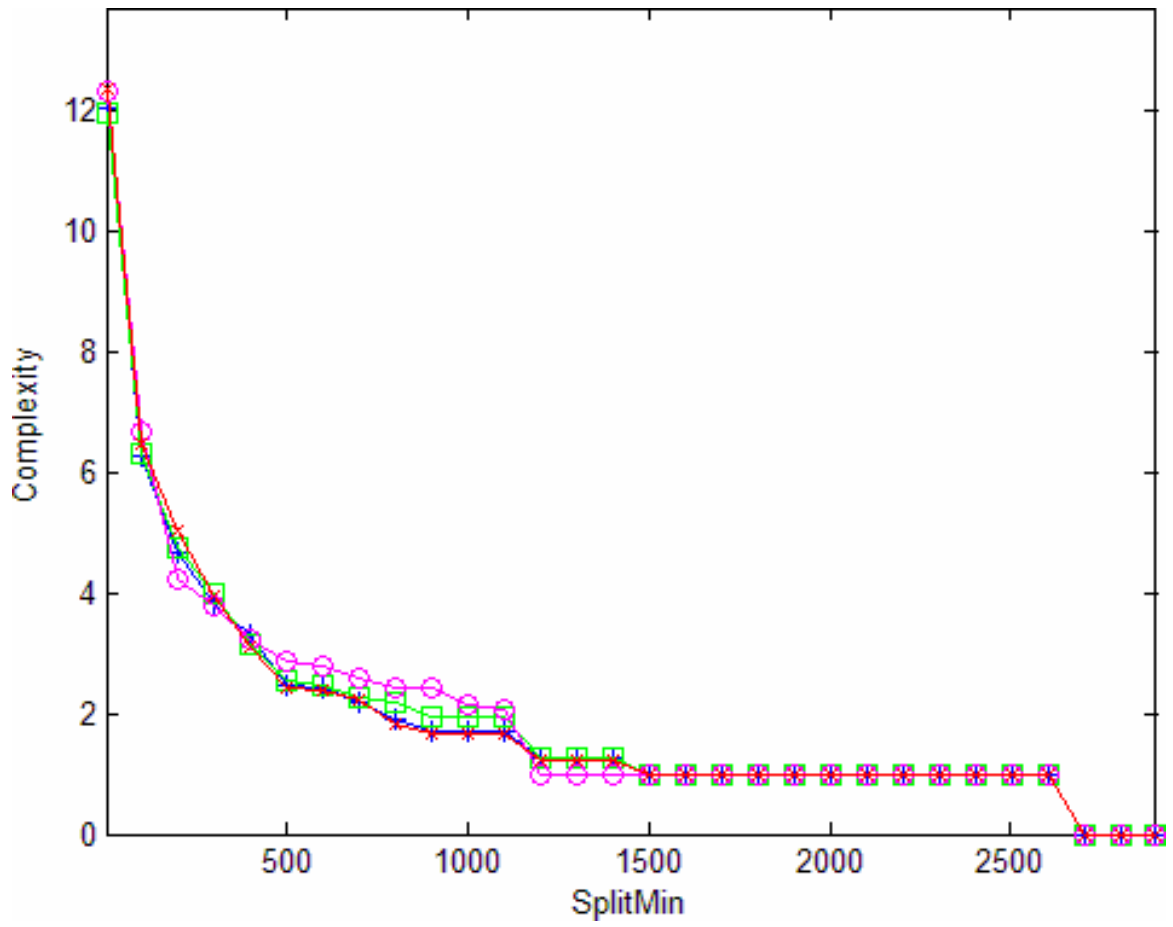
(h)

despite the large true positive counts, MDR also produces large false negative counts, whereas the Gini Index, Twoing Rule, and Rough Product do not in this splitmin range. That is, the high accuracy of the Gini Index, Twoing Rule, and Rough Product is predominantly due to large true negative counts.

In terms of mean node count, mean tree depth, and mean complexity, the structural metric results, shown in Figure 4.12(f, g, h), show no significant differences between the splitting criteria. The Rough Product splitting criterion seems to follow the structure of the Gini Index splitting criterion more closely than the other splitting criteria.

4.2.2.3   United States Income Data Set

In this experiment, we use United State Census Bureau data from the University of California Irving Machine Learning Repository. The data contains 48,842 U.S. citizens, each containing six numerical attributes and eight categorical attributes, for the purposes of classifying citizens that have an income less than $50,000. We randomly chose 497 samples from the original data set. Using the entire original data set is unnecessary since our objective is to evaluate the splitting criteria and not produce the best performing classifiers.

Our experiment uses 10-folds cross validation to evaluate the splitting criteria. We cover splitmins ranging from 1 to 500 at a step interval of five, and provide the same training and testing sets without biases or cost functions to each tree-growing algorithm.

The classification metric results, shown in Figure 4.13(a, b, c, d, e), clearly show that the Rough Product splitting criterion outperforms the Gini Index, Twoing Rule, and MDR splitting criteria. However, the Rough Product's increased classification accuracy also causes an increase in structural size and complexity, as shown in Figure 4.13(g, h, i). Despite this result, the Rough Product splitting criterion is still preferred for this data set because the Gini Index, Twoing Rule, and MDR splitting criteria do not produce trees that substantially classify better than blindly guessing the most common decision value.

Conclusion

In this chapter, we evaluated the Gini Index, Twoing Rule, Maximum Deviance Reduction, and Rough Product splitting criteria on various data sets. We performed k-folds cross validation using the EvalTreeGUI tool to measure several classification and structural metrics. The experiments showed that in data with low noise between decision classes, the Rough Product splitting criterion generally performs as well as the Gini Index or Twoing Rule splitting criteria across all metrics. In terms of the classification metrics, our experiment with the U.S. Census Income data set showed that the Rough Product splitting criterion produces meaningful splits when the Gini Index, Twoing Rule, and MDR splitting criteria do not. In terms of tree structure, the Gaussian Experiments showed that the Rough Product splitting criterion provides strikingly large reductions in node count, tree depth and complexity in response to increased noise between decision classes. This result suggest that, in the presence of noisy data, the Rough Product splitting criterion could construct binary decision trees

140

that are simpler and shorter than those produced by the Gini Index, Twoing Rule, or

Maximum Deviance Reduction splitting criteria. Although our study does not presume

the superiority of any single splitting criterion, we conclude that the Rough Product

splitting criterion is a viable option for binary decision tree construction.
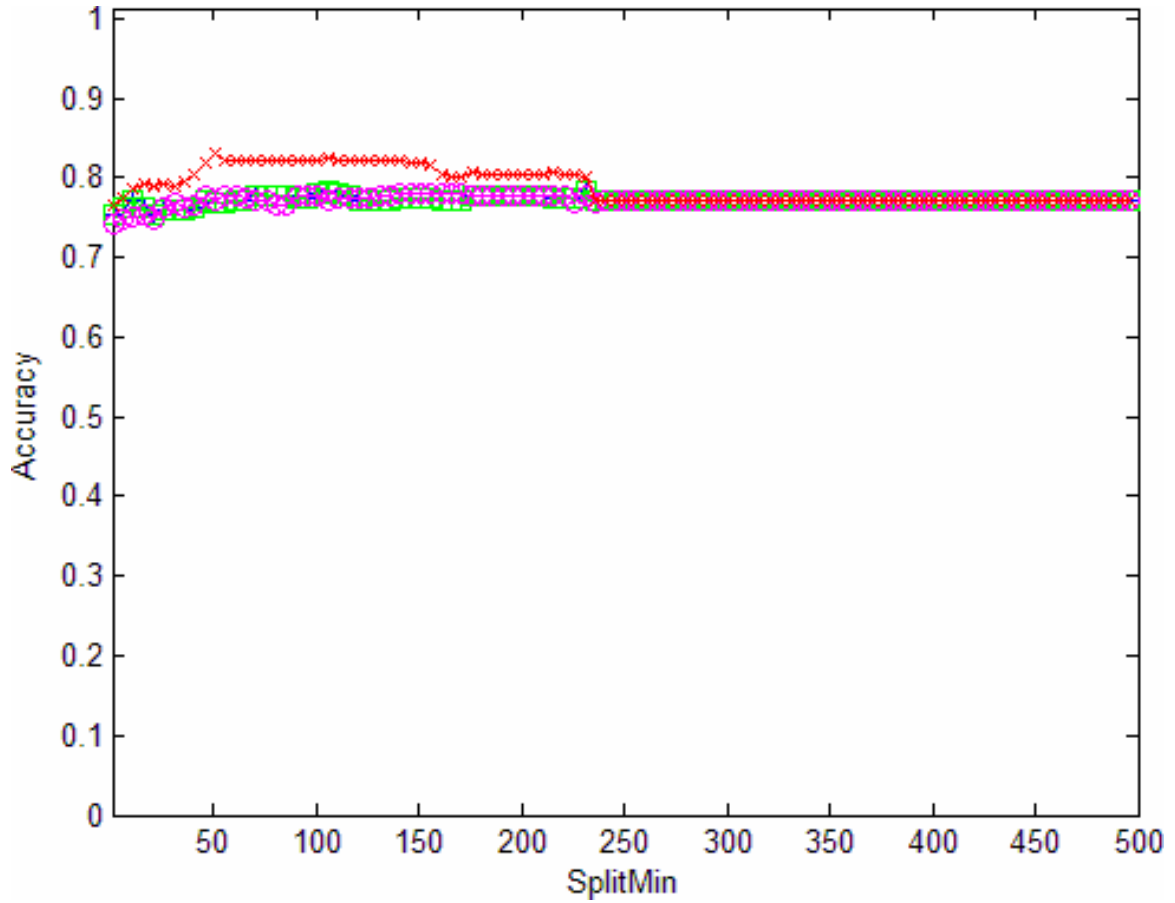
(a)

Figure 4.13: United States Income Data Set. Provided are the fold mean results for the mean accuracy (a), mean error rate (b), mean precision (c), mean recall (d), mean F-measure (e), mean node count (f), mean tree depth (g), and mean complexity (h) for the Gini Index (blue crosses), Twoing Rule (green square), Maximum Deviance Reduction (magenta circles) and Rough Product (red 'x's) splitting criteria. (This figure is presented in color; the black and white reproduction may not be an accurate representation.)

Figure 4.13 – Continued
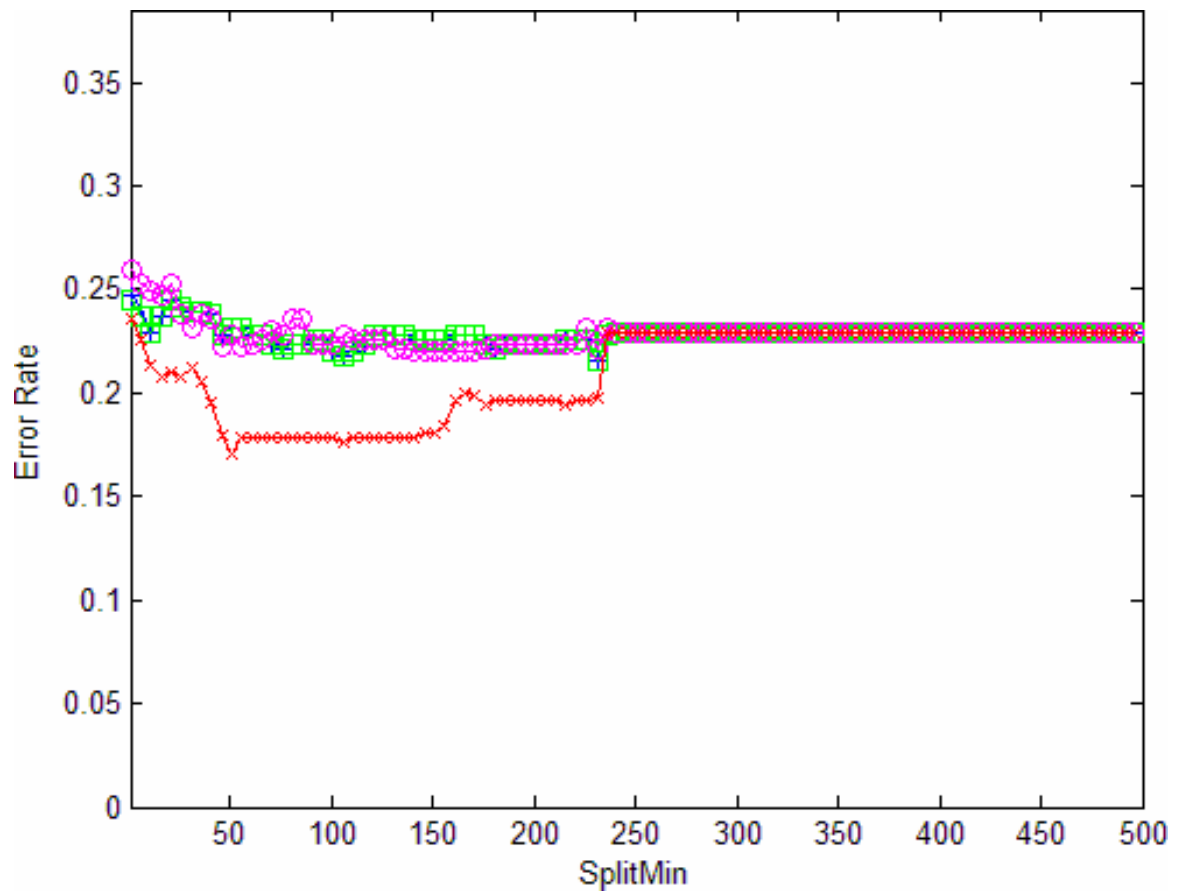
(b)

Figure 4.13 – Continued
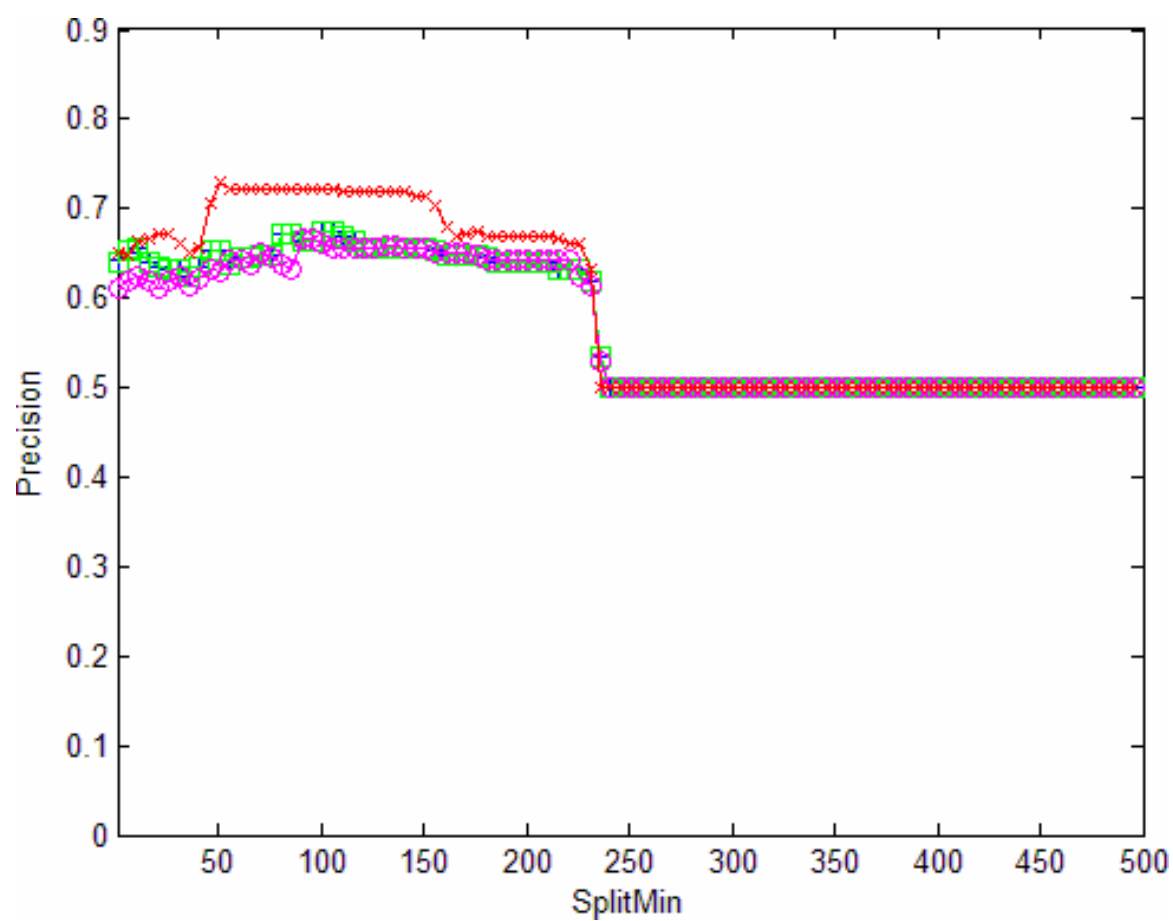
(c)

Figure 4.13 – Continued

(d)

145
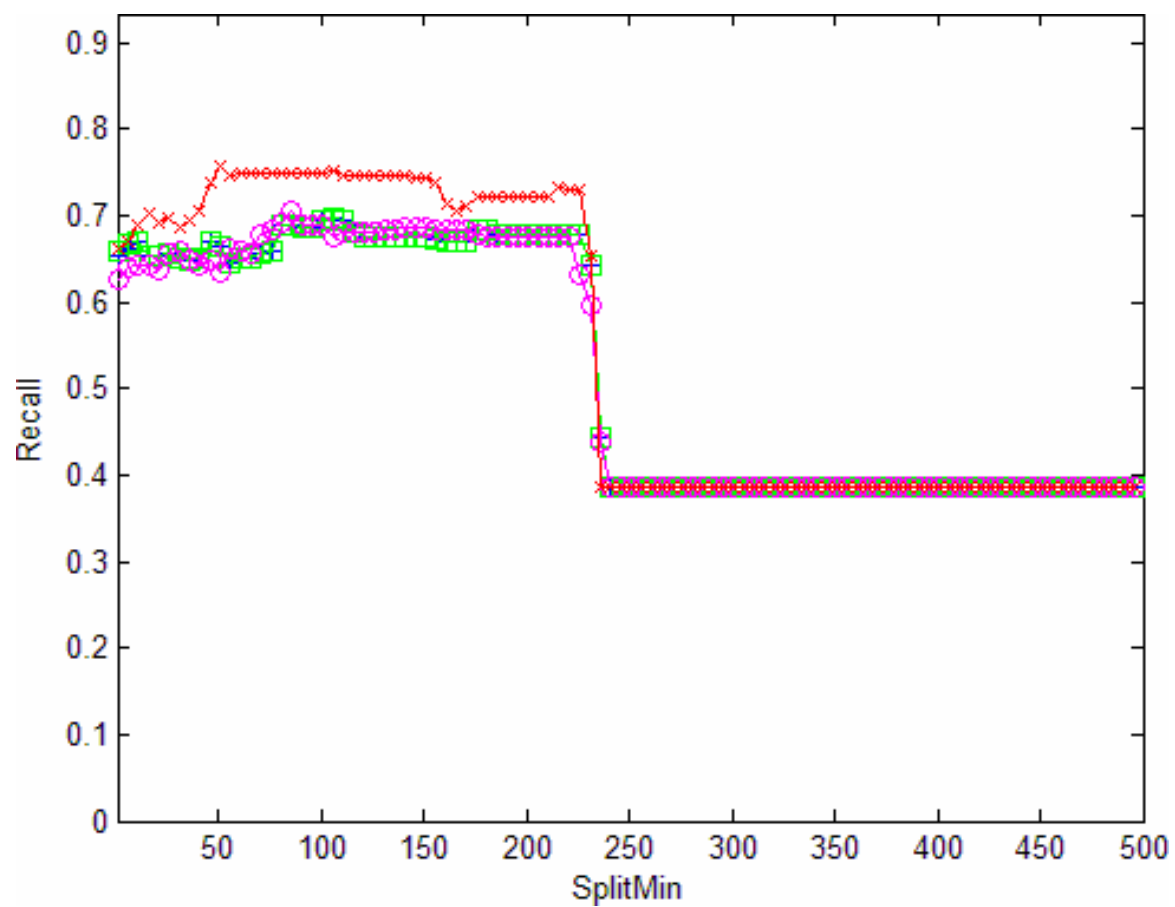
Figure 4.13 – Continued

(e)

146
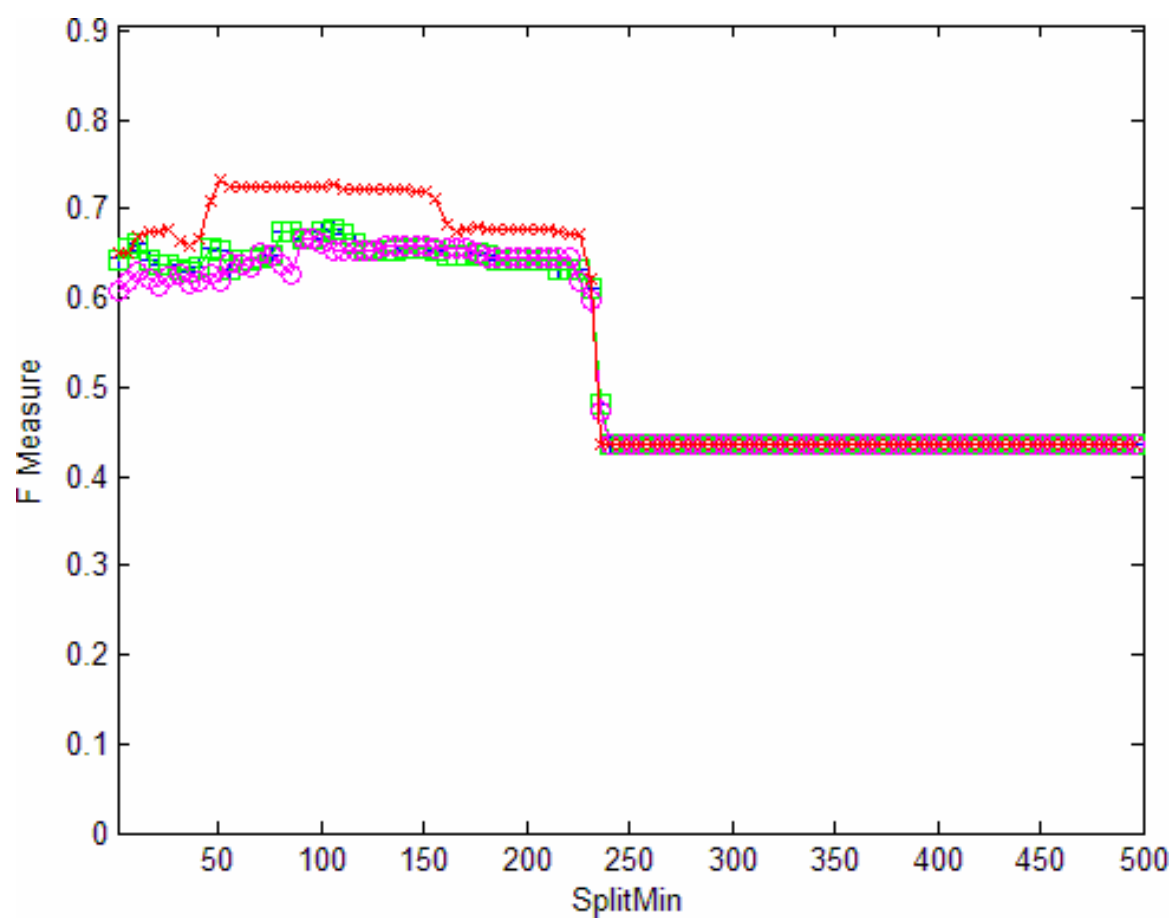
Figure 4.13 – Continued
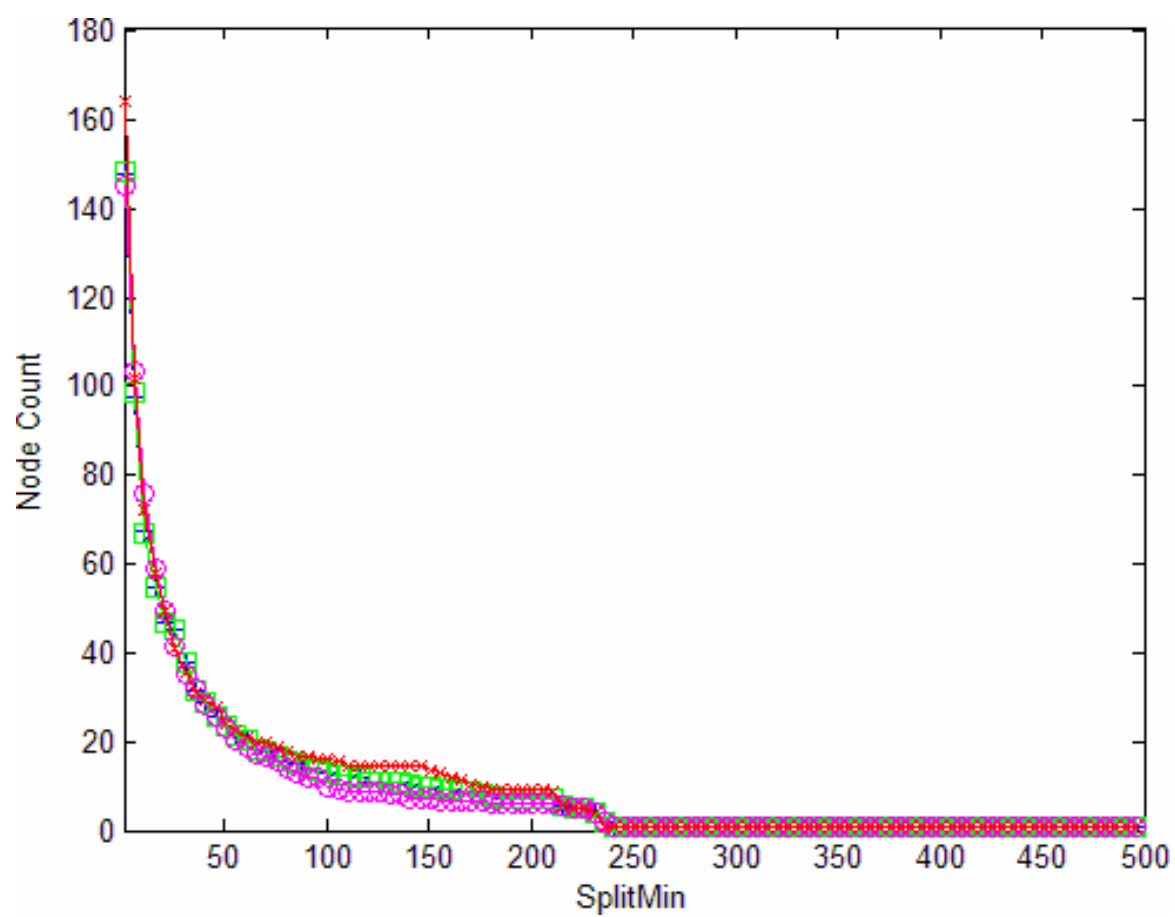
(f)

Figure 4.13 – Continued
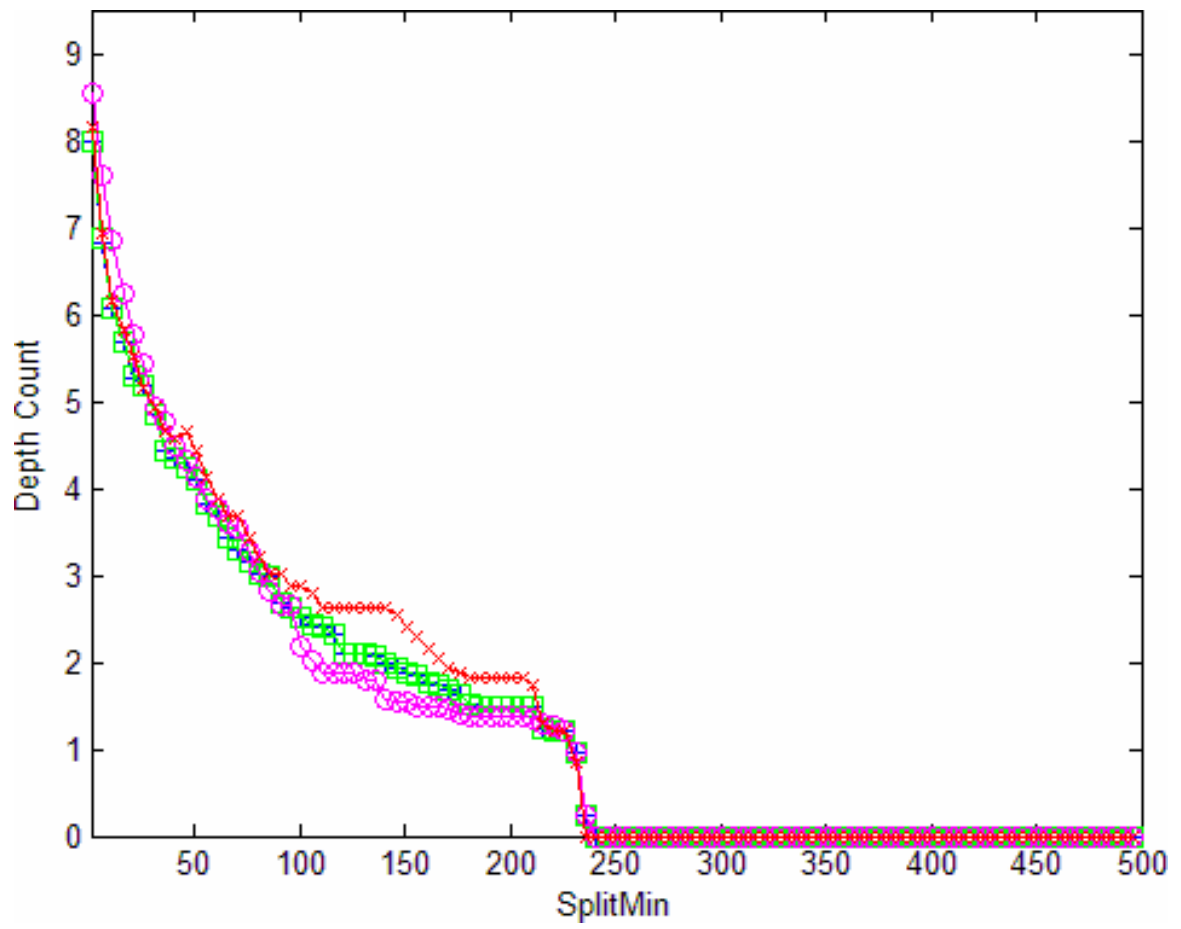
(g)

Figure 4.13 – Continued
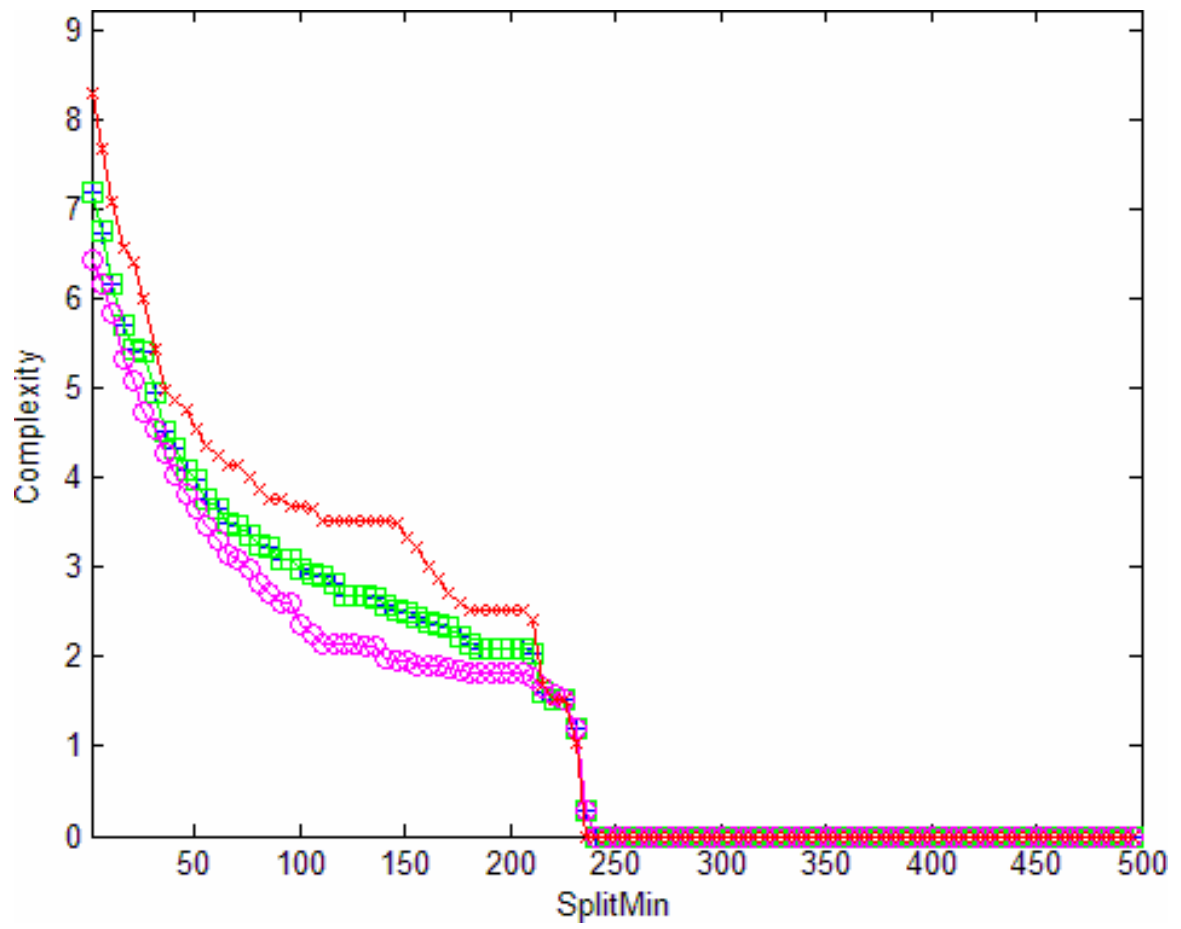
(h)

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORK

Conclusions

Our research aimed to advance induction techniques for pattern classification using rough set theory as a basis. To narrow our scope in this broad objective, we chose to work with binary decision tree classifiers. We saw that decision tree optimization is in the class of NP-complete [17], meaning it requires exponential time complexity in a deterministic algorithm to ensure an optimal decision tree structure. Therefore, numerous researchers have developed heuristics in an effort to determine near-optimal decision trees in polynomial time. We reviewed various preprocessing methods, splitting criteria, stopping criteria, and pruning methods typically applied to decision tree construction. In addition, we reviewed rough set theory and several algorithms that use it for feature selection. We then proposed a new measure of information based on rough set theory (which we call the *rough product*) to help us understand the manner in which an attribute value partition will affect the upper approximation for each decision class. We subsequently applied the rough product in a splitting criterion for a binary decision tree classifier. We evaluated this criterion against the Gini Index, Twoing Rule, and Maximum Deviance Reduction splitting criteria using *k*-folds cross validation. Using several classification and structural metrics, we compared the performance of each splitting criterion on artificial Gaussian distribution cluster data sets and real-world data sets. For the Gaussian clusters, we

150

observed that the Rough Product splitting criterion provided strikingly large reductions in tree depth and complexity in response to increased noise between decision classes. For the real-world data sets, we observed that the Rough Product splitting criterion generally performed as well as or better than competing splitting criteria for all metrics. We conclude that the Rough Product measure of information has the potential to improve induction in classification models. In particular, we conclude that the Rough Product splitting criterion provides a viable option for binary decision tree construction. The empirical results suggest that, in the presence of noisy data, the Rough Product splitting criterion could be useful in constructing simpler, shorter trees than the Gini Index, Twoing Rule, or Maximum Deviance Reduction splitting criteria.

Future Work

The objective for future endeavors will remain the same – to improve induction for pattern recognition applications. The work completed in this thesis will serve as the root for these endeavors that may potentially branch out in the following directions:

1. Study the Rough Product splitting criterion for binary decision trees with more artificially generated and real-world data sets.

2. Search for strengths and weaknesses in the proposed Rough Product measure of information and derive ways to improve the weaknesses.

3. Extend the binary Rough Product splitting criterion into an *n-ary* Rough Product splitting criterion.

4. Apply and study the effectiveness of the Rough Product as the information function in various information-based algorithms.

151

APPENDIX

EXPERIMENTAL RESULTS SUMMARY TABLES

Table A.1

Mean of Mean Accuracy (MMA) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.9201	0.9201	0.9201	0.9229
Gaussian 2a	0.8818	0.8819	0.8777	0.883
Gaussian 2b	0.8943	0.8943	0.8912	0.8943
Gaussian 3a	0.7110	0.7111	0.7030	0.7078
Gaussian 3b	0.6712	0.6712	0.6692	0.6720
Gaussian 4a	0.4902	0.4915	0.4872	0.5090
Gaussian 4b	0.4955	0.4955	0.4925	0.5048
Breast Cancer	0.9016	0.9016	0.8938	0.9016
Five Cancers	0.7825	0.7826	0.7836	0.7819
Income	0.7713	0.7713	0.771	0.7887

Note: The table shows that the Rough Product splitting criterion performed, on average, as well as or marginally better than the Gini Index, Twoing Rule, and MDR splitting criteria. Note that in the Income experiment, the Rough Product performed noticeably better. We also see that the Gini Index and Twoing Rule criteria have practically equivalent MMA for most data sets.

Table A.2

Mean of Mean Error Rate (MMER) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.0799	0.0799	0.0799	0.0771
Gaussian 2a	0.1182	0.1181	0.1223	0.1170
Gaussian 2b	0.1057	0.1057	0.1088	0.1057
Gaussian 3a	0.2890	0.2889	0.2970	0.2922
Gaussian 3b	0.3288	0.3288	0.3308	0.3280
Gaussian 4a	0.5098	0.5085	0.5128	0.4910
Gaussian 4b	0.5045	0.5045	0.5075	0.4952
Breast Cancer	0.0984	0.0984	0.1062	0.0984
Five Cancers	0.2175	0.2165	0.2164	0.2181
Income	0.2287	0.2287	0.229	0.2113

Note: The table shows that the Rough Product splitting criterion performed, on average, as well as or marginally better than the Gini Index, Twoing Rule, and MDR splitting criteria. Note that in the Income experiment, the Rough Product performed noticeably better. We also see that the Gini Index and Twoing Rule criteria have practically equivalent MMA for most data sets.

Table A.3

Mean of Mean Precision (MMP) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.9291	0.9291	0.9291	0.9328
Gaussian 2a	0.8925	0.8926	0.8878	0.8942
Gaussian 2b	0.8998	0.8998	0.8967	0.8999
Gaussian 3a	0.7128	0.7128	0.7034	0.7093
Gaussian 3b	0.6758	0.6759	0.6736	0.6759
Gaussian 4a	0.4960	0.4966	0.4940	0.5129
Gaussian 4b	0.5035	0.5034	0.5010	0.5114
Breast Cancer	0.8839	0.8839	0.8818	0.8836
Five Cancers	0.3996	0.4038	0.4154	0.4003
Income	0.5701	0.5701	0.5668	0.5897

Note: The table shows that the Rough Product splitting criterion performed, on average, as well as or marginally better than the Gini Index, Twoing Rule, and MDR splitting criteria. Note that in the Gaussian 4a, Gaussian 4b, and Income experiments, the Rough Product performed noticeably better. We also see that the Gini Index and Twoing Rule criteria have practically equivalent MMP for most data sets.

Table A.4

Mean of Mean Recall (MMR) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.8934	0.8934	0.8934	0.8956
Gaussian 2a	0.8542	0.8543	0.8506	0.8546
Gaussian 2b	0.8645	0.8645	0.8614	0.8646
Gaussian 3a	0.7246	0.7246	0.7176	0.7182
Gaussian 3b	0.6452	0.6452	0.6440	0.6453
Gaussian 4a	0.4583	0.4604	0.4543	0.4757
Gaussian 4b	0.4699	0.4699	0.4673	0.4769
Breast Cancer	0.8596	0.8596	0.8488	0.8599
Five Cancers	0.2656	0.2681	0.2723	0.2624
Income	0.5203	0.5204	0.519	0.5456

Note: The table shows that the Rough Product splitting criterion performed, on average, as well as or marginally better than the Gini Index, Twoing Rule, and MDR splitting criteria. Note that in the Gaussian 4a, Gaussian 4b, and Income experiments, the Rough Product performed noticeably better. We also see that the Gini Index and Twoing Rule criteria have practically equivalent MMR for most data sets.

Table A.5

Mean of Mean F-Measure (MMFM) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.9035	0.9035	0.9035	0.9065
Gaussian 2a	0.8637	0.8638	0.8594	0.865
Gaussian 2b	0.8755	0.8755	0.8723	0.8755
Gaussian 3a	0.7024	0.7025	0.6935	0.7007
Gaussian 3b	0.6494	0.6495	0.6469	0.6500
Gaussian 4a	0.4582	0.4588	0.4537	0.4763
Gaussian 4b	0.4644	0.4644	0.4614	0.4806
Breast Cancer	0.8669	0.8669	0.8596	0.8669
Five Cancers	0.3028	0.3059	0.3117	0.3001
Income	0.5359	0.5359	0.5323	0.5577

Note: The table shows that the Rough Product splitting criterion performed, on average, as well as or marginally better than the Gini Index, Twoing Rule, and MDR splitting criteria. Note that in the Gaussian 4a, Gaussian 4b, and Income experiments, the Rough Product performed noticeably better. We also see that the Gini Index and Twoing Rule criteria have practically equivalent MMFM for most data sets.

Table A.6

Mean of Mean Node Count (MMNC) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	2.742	2.742	2.742	2.742
Gaussian 2a	4.935	4.935	4.452	4.929
Gaussian 2b	7.6	7.6	7.039	7.587
Gaussian 3a	17.63	17.64	17.37	16.63
Gaussian 3b	41.23	41.23	40.32	40.53
Gaussian 4a	18.82	18.8	21.69	14.81
Gaussian 4b	77.62	77.37	83.41	65.66
Breast Cancer	7.52	7.52	6.006	7.463
Five Cancers	78.84	78.81	78.95	80.61
Income	10.49	10.52	10	11.61

Note: The table shows the Rough Product splitting criterion correlating strongly to the Gini Index and Twoing Rule splitting criteria in data sets with low noise. However, as the noise increases, the Rough Product MMNC increases more slowly than the MMNC for the Gini Index and Twoing Rule splitting criteria. In fact, we see that the Rough Product splitting criterion producing significantly lower MMNCs for Gaussian Experiments 4a and 4b against all splitting criteria. This result suggests that the Rough Product splitting criterion may be useful in ensuring simpler trees in the presence of noisy data.

Table A.7

Mean of Mean Depth Count (MMDC) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.5806	0.5806	0.5806	0.5806
Gaussian 2a	0.9832	0.9832	0.9195	0.9819
Gaussian 2b	1.247	1.247	1.178	1.246
Gaussian 3a	2.259	2.259	2.305	2.0960
Gaussian 3b	1.431	1.431	1.440	1.356
Gaussian 4a	2.617	2.620	3.211	1.644
Gaussian 4b	4.395	4.394	5.731	1.639
Breast Cancer	1.425	1.425	1.117	1.417
Five Cancers	1.685	1.709	1.757	1.676
Income	1.355	1.356	1.34	1.484

Note: The table shows the Rough Product splitting criterion correlating strongly to the Gini Index and Twoing Rule splitting criteria in data sets with low noise. However, as the noise increases, the Rough Product MMDC increases more slowly than the MMDC for the Gini Index and Twoing Rule splitting criteria. In fact, we see the Rough Product splitting criterion producing significantly lower MMDCs for Gaussian Experiments 4a and 4b against all splitting criteria. In addition, the MMDC results between Gaussian Experiments 4a and 4b for the Rough Product splitting criterion show no change. This result may suggest that the Rough Product splitting criterion could be useful in ensuring shorter trees in the presence of noisy data. The result in the Five Cancers data set supports this hypothesis.

Table A.8

Mean of Mean Complexity (MMC) for the Fold Mean.

Experiment	Gini Index	Twoing Rule	MDR	Rough Product
Gaussian 1	0.871	0.871	0.871	0.871
Gaussian 2a	1.376	1.376	1.260	1.372
Gaussian 2b	1.784	1.784	1.627	1.782
Gaussian 3a	2.737	2.738	2.596	2.585
Gaussian 3b	1.731	1.731	1.671	1.661
Gaussian 4a	3.378	3.381	4.085	2.012
Gaussian 4b	5.439	5.439	7.175	2.035
Breast Cancer	1.91	1.91	1.402	1.909
Five Cancers	2.004	2.041	2.083	2.016
Income	1.512	1.513	1.323	1.791

Note: The table shows the Rough Product splitting criterion correlating strongly to the Gini Index and Twoing Rule splitting criteria in data sets with low noise. However, as the noise increases, the Rough Product MMC increases more slowly than the MMC for the Gini Index and Twoing Rule splitting criteria. In fact, we see the Rough Product splitting criterion producing significantly lower MMCs for Gaussian Experiments 4a and 4b against all splitting criteria. In addition, the MMC results between Gaussian Experiments 4a and 4b for the Rough Product splitting criterion show practically no change. This result may suggest that the Rough Product splitting criterion could be useful in ensuring faster classification than other splitting criteria, especially when presented with noisy data.

REFERENCES

[1]   Aboul-Hassan S. and Bouchaffra D. (2001). Automotive Design Driven by Pattern Recognition. In: *Proceedings of the Artificial Neural Network in Engineering (ANNIE'2001) Conference*. University of Missouri-Rolla. 4-7 November 2001.

[2]   Bohn C. (1997). An Incremental Unsupervised Learning Scheme for Function Approximation. In: *Proceedings of the 1997 IEEE International Conference on Neural Networks (ICNN'97)*. Piscataway, NJ: IEEE Service Center. pp. 1792-1797.

[3]   Bouchaffra D., Govindaraju V., and Srihari S.N. (1999). Recognition of Strings using Non-stationary Markovian Models: An Application to ZIP Code Recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*. Fort Collins, Colorado, USA. 23-25 June 1999.

[4]   Bouchaffra D. and Tan J. (2006). Introduction to Structural Hidden Markov Models: Application to Handwritten Numeral Recognition. In Famili A. eds.: *Intelligent Data Analysis Journal*. IOS Press. 10(1).

[5]   Bouchaffra D. and Tan J. (2006). Structural Hidden Markov Models using a Relation of Equivalence: Application to Automotive Designs. In Webb G. eds.: *Data Mining and Knowledge Discovery Journal*. 12(1): 79-96.

[6]   Bouchaffra D. and Tan J. (2006). Protein Fold Recognition using a Structural Hidden Markov Model. In: *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*. Hong Kong. 20-24 August 2006.

[7]   Breiman L., Friedman J., Olshen R., and Stone C. (1993) *Classification and Regression Trees*. New York: Chapman & Hall.

[8]   Brudnak M. (2005). Support Vector Methods for the Control of Unknown Nonlinear Systems. *Ph.D. Thesis.* Oakland University, Rochester MI, USA.

[9]   Capelle A.S., Alata O. Fernandez-Maloigne C., and Ferrie J.C. (2001). Unsupervised Algorithm for the Segmentation of Three-Dimensional Magnetic Resonance Brain Images. In: *Proceedings for the International Conference on Image Processing (ICIP'01)*. 7-10 Oct 2001. 3: pp. 1047-1050.

[10] Clark A. (2001). Unsupervised Induction of Stochastic Context-Free Grammars Using Distributional Clustering. In Daelemans W., Zajac R.m eds.*: Proceedings of 5th Conference on Natural Language Learning (CoNLL-2001)*. Toulouse, France. pp. 105-112.

[11] Cohen P., Heeringa B., and Adams N.M. (2002). An Unsupervised Algorithm for Segmenting Categorical Time-series into Episodes. In Hand D.J., Adams N.M., and Bolton R.J., eds.: *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery (September 16-19, 2002)*. London: Springer-Verlag. LNCS 2447: pp. 49-62.

[12] Duda R., Hart P., and Stork D. (2001). *Pattern Classification*. 2nd ed. Canada: John Wiley & Sons, Inc. pp. 1-19, 394-413, 454-458, 482-485.

[13] Duntsch I. and Gunther, G. (2000). Rough Set Data Analysis. *Encyclopedia of Computer Science and Technology*. 43: pp. 281-301.

[14] Esposito F., Maberba D., Semeraro G., and Tamma V. (1999). The Effects Of Pruning Methods On The Predictive Accuracy Of Induced Decision Trees. *Applied Stochastic Models In Business And Industry*. 15: pp. 277-299.

[15] Herbert J. and Yao J. (2005). Time-Series Data Analysis with Rough Sets. *4th International Conference on Computational Intelligence in Economics and Finance (CIEF)*. Salt Lake City, UT, USA. 21-26 July 2005. pp. 908-911.

[16] Hu X. (1995). Knowledge Discovery in Databases: An Attribute-Oriented Rough Set Approach. *Ph.D. Thesis*. University of Regina. Regina, Saskatchewan, Canada.

[17] Hyafil L. and Rivest R. (1976). Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letters*. 5(1).

[18] Kantardzic, M. (2003). Preparing The Data. In: Kartalopoulos V. *Data Mining Concepts, Models, Methods, and Algorithms*. Piscataway, NJ, USA: Wiley-IEEE Press. pp. 19-38.

[19] Karlsen R., Gorsich D., and Gerhart G. (2000). Support Vector Machines and Target Classification. In: *Proceedings of the 22nd Army Science Conference*. Baltimore MD, USA. pp. 874-881.

[20] Karlsen R., Gorsich D., and Gerhart G. (2001). Target Acquisition and Human Vision Modeling. In: *Proceedings of the 12th Ground Vehicle Survivability Symposium*. Monterey CA.

[21] Kohavi R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. pp. 1137-1143.

[22] Lixiang S. (2001). Data Mining Techniques Based on Rough Set Theory. *Ph.D. Thesis*, National University of Singapore, Singapore.

[23] Modrzejewski M. (1993). Feature Selection using Rough Sets Theory. In Brazdil P.B. eds.: *Proceedings of the European Conference on Machine Learning*. Springer-Verlag. pp. 213-226.

[24] Milde H., Hotz L., Kahl J., Neumann B., and Wessel S. (1999). MAD: A Real World Application of Qualitative Model-Based Decision Tree Generation for Diagnosis. In: *Proceedings of Industrial and Engineering Applications for Artificial Intelligence and Expert Systems*. pp. 246-255.

[25] Moshkov M. (2005). Time Complexity of Decision Trees. In Peters J.F. and Skowron A eds.: *Transactions on Rough Sets III*. New York: Springer-Verlag. LNCS 3400, pp. 244-459.

[26] Murthy S. (1998). Automatic Construction of Decision Trees From Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*. Boston: Kluwer Academic Publishers. 2(4): pp. 345-389.

[27] Nguyen H.S. and Nguyen S.H. (1999) An Application of Discretization Methods in Control. *citeseer.ist.psu.edu/son99application.html*

[28] Oliver, N.M., Rosario B., and Pentland A. (2000). A Bayesian Computer Vision System for Modeling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 22(8): pp. 831-843.

[29] Pawlak Z. (1982). Rough Sets. *International Journal of Computer Information Science*. 11: pp. 341-356.

[30] Pawlak Z. (2002). A Primer on Rough Sets - A New Approach To Drawing Conclusions From Data. *Cardozo Law Review*. 22: pp. 1407-1415.

[31] Pawlak Z. (2005). A Treatise on Rough Sets. In Peters J.F. and Skowron A eds.: *Transactions on Rough Sets IV*. Berlin: Springer-Verlag. LNCS 3700: pp. 1-17.

[32] Pechyony D. (2004). Decision Trees: More Theoretical Justification For Practical Algorithms. *Masters Thesis*. Tel-Aviv University. Tel-Aviv, Israel.

[33] Quinlan J.R. (1986). Induction of decision trees. *Machine Learning*. 1(1): 81-106.

[34] Quinlan J.R. and Rivest R.L. (1989). Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*. 80(3): pp. 227-248.

[35] Quinlan J.R. (1993). C4.5: Programs for Machine Learning. San Francisco: Morgan Kaufmann Publishers.

[36] Rajagopalon A.N. and Chellappa R. (2000). Vehicle Detection and Tracking in Video. In: *Proceedings of the International Conference on Image Processing (ICIP'00)*. Vancouver, BC, Canada. 1: pp. 351-354.

[37] Safavian S. and Landgrebe D. (1991). A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*. 21(3): pp. 660-674.

[38] Shannon C.E. (1948). A Mathematical Theory of Communication. Reprinted with corrections from *The Bell System Technical Journal*, 27: pp. 379–423.

[39] Shen Q. and Chouchoulas A. (2001). Rough Set-Based Dimensionality Reduction For Supervised and Unsupervised Learning. *International Journal of Applied Mathematics and Computer Science*. 11(3): pp. 583-601.

[40] Shoshani L. and Bouchaffra D. (2001). Identification of Handwritten Digits Using K-Nearest Neighbor. In: *Proceedings of the 2nd IEEE Electro-Information Technology Conference*. Oakland University. 7-9 June 2001.

[41] Sipser M. (1997). Time Complexity. In: *Introduction To The Theory Of Computation*. Boston, MA: PWS Publishing Company. pp. 225-276.

[42] Teutsch C., Berndt D., Trostmann E., and Weber M. (2006). Real-Time Detection of Elliptic Shapes for Automated Object Recognition and Object Tracking. In: *Proceedings of Machine Vision Applications in Industrial Inspection XIV*. Bellingham, Washington, USA. 15-19 January 2006. pp. 171-179.

[43] Wei J., Wang S., Ma Z., and Huang D. (2002). Rough set based decision tree. In: *Proceedings of the 4th World Congress on Intelligent Control and Automation*. pp. 426-431.

[44] Wei J. (2003). Rough Set Based Approach to Selection of Node. *International Journal of Computational Cognition*. 1(2): pp. 25-40.

[45] Yao J.T. and Zhang M. (2005). Feature Selection with Adjustable Criteria. In: *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC'05)*. Regina, Saskatchewan, Candada. 1-3 September 2005. LNAI 3641: pp.204-213.

[46] Zhang M. and Yao J.T. (2004). A Rough Sets Based Approach to Feature Selection. In: *Proceedings of the 23rd International Conference of NAFIPS*. Banff, Alberta, Canada. 27-30 June 2004. pp. 434-439.

[47] Zhong N., Dong J.Z., and Ohsuga, S. (2001).Using Rough Sets with Heuristics for Feature Selection. *Journal of Intelligent Information Systems*. 16: pp. 199-214.

[48] Zhong N. and Skowron A. (2001). A Rough Set-Based Knowledge Discovery Process. *International Journal of Applied Mathematics and Computer Science*. 11(3): pp. 603-619.

[49] Zhou Q. (2003). Adaptive Knowledge Discovery Techniques for Data Mining, *Ph.D. Thesis*. University of Otago. Dunedin, New Zealand.

[50] Zohdy M., Bouchaffra D., and Quinlan J. (2001). Optimal Mapping from Chromosome Space to Feature Space for Solving Sequential Pattern Recognition Problems. In: *Proceeding for the IEEE Midwest Symposium on Circuit and Systems*. Dayton, Ohio, USA. 14-17 August 2001.

[51] Zorman M., Eich H., Stiglic B., Ohmann C., and Lenic M. (2002). Does Size Really Matter – Using a Decision Tree Approach for Comparison of Three Different Databases from the Medical Field of Acute Appendicitis. *Journal of Medical Systems*. 26(5): pp. 465-477.